(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 April 2001 (12.04.2001)

PCT

(10) International Publication Number
WO 01/25873 A2

(51) International Patent Classification[7]: G06F

(21) International Application Number: PCT/US00/27191

(22) International Filing Date: 3 October 2000 (03.10.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/414,402 7 October 1999 (07.10.1999) US

(71) Applicant: SUN MICROSYSTEMS, INC. [US/US]; 901 San Antonio Road, M/S: UPAL01-521, Palo Alto, CA 94303 (US).

(72) Inventors: DIGIORGIO, Rinaldo; 20 Mile Common Road, Easton, CT 06612 (US). UHLER, Stephen; Mundell Way, Los Altos, CA 94022 (US).

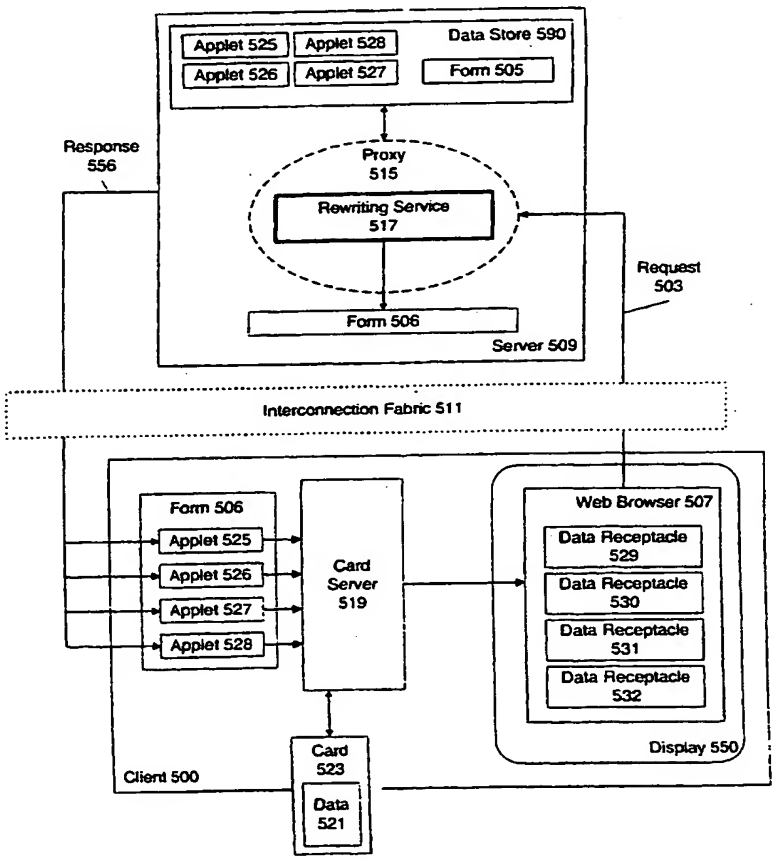
(74) Agents: HECKER, Gary, A. et al.; The Hecker Law Group, 1925 Century Park East, Suite 2300, Los Angeles, CA 90067 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR COMPLETING A FORM



(57) Abstract: The present invention provides the user with a mechanism for completing a form. When a user issues a request for a form, that request is processed by a rewriting service. In one embodiment of the invention, the rewriting service provides the user with a mechanism for providing data to a form by obtaining data from a data source location. The rewriting service responds to a request for a form by generating a version of the form that contains embedded programs (e.g. applets). The applet elements identify the location of applets configured to obtain data from the data source location. A smart card is an example of a data source location. Access to the smart card may be controlled by the use of an authentication mechanism. The rewritten form is transmitted to a client computer where it utilizes the embedded programs to obtain data from a data source location such as a smart card.

WO 01/25873 A2

Published:
— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

METHOD AND APPARATUS COMPLETING A FORM

BACKGROUND OF THE INVENTION

1. FIELD OF THE INVENTION

The present invention relates to computer networks and more particularly to a method and apparatus for completing a form.

2. BACKGROUND ART

Computer users can shop for merchandise and/or information using the Internet. A number of companies maintain web sites where customers can purchase books, records, videos, and many other products. Other web sites provide subscription services that provide the user with access to a service or product, such as channels. Typically, users make purchases or sign up for services by completing forms. A problem with current Internet schemes is that each form must be filled out separately and manually. This is a time consuming and often frustrating process for the user. The product that do allow the user to automatically complete forms do not do so in a secure manner (e.g. IES). For example, mechanisms which do allow a form to be automatically completed if the user visits a form twice (e.g. cookies) are not secure. Such mechanisms also do not allow the user to automatically complete forms from more than one location. Additionally, such mechanisms do not provide users with a way to physically control access to data that is utilized to complete a form.

Current systems do not provide a way for users to quickly complete a form (referred to as "populating" a form) with data. For example, when a user wishes to purchase a product online, the user typically supplies a credit card

number, a shipping address, and any other personal information that may be needed to complete the transaction. The user supplies this information using an input device such as a keyboard to manually enter data into the fields of a form. However, after buying goods from one vendor, for example, the user may wish to purchase goods from a second vendor. The user must manually complete the second vendor's form even though the second vendor may require the purchaser to provide the same kind of information the user provided to the first vendor. This presents a problem to the user because it forces the user to manually enter the same information into different vendor's forms multiple times. Some prior art mechanisms utilize a scheme that provides repeat users with a way to fill out a form with information already provided. However, these schemes do not allow the user to personally control what information is provided to the vendor. Moreover, such techniques do not allow the user to specify how long the data may be kept.

Another problem presented by the prior art is that users do not have a way to securely store sensitive information and then later use that information to populate a form. Current systems do not provide a way to prevent unauthorized users from obtaining access to sensitive information stored on a user's computer. Instead such systems leave data that is entered into a form readily accessible. For example, data that is entered into a form may be stored locally using a technology called cookies. A cookie is a local representation of information related to a particular web page that was previously visited by the user. Cookies are not encrypted by default and may be read by anyone who can obtain them. This is problematic because it unnecessarily exposes sensitive information to unauthorized users. Therefore, there is a need for a method and apparatus for completing a form with data where that data is accessible only to authenticated users.

The problems associated with form population may be better understood by the following discussion of the Internet/World Wide Web, web page creation, embedded forms, cookies, and web browser technology.

The Internet / World Wide Web:

A web browser is a type of computer program that provides users with a mechanism for accessing the World Wide Web (WWW). The WWW is a segment of the Internet comprised of numerous web clients and web servers that communicate with one another using a standard set of protocols. A web server is a computer configured to provide web pages to the web client upon request. A web client typically utilizes the web browser application to request web pages from the web server.

The Internet is a global computer network comprised of an amalgamation of interconnected networks that are capable of readily communicating with one another using a standardized set of protocols. Protocols provide a uniform set of communication parameters that enable computers to effectively transmit and receive data. Most computer networks, including the Internet, utilize several different layers of protocols. Each layer provides the network with different functionality.

The WWW is a segment of the Internet that utilizes an application layer protocol called the HyperText Transfer Protocol (HTTP) to disseminate and to obtain information from users. HTTP is a request/response protocol used with distributed, collaborative, hypermedia information systems. In operation, HTTP enables one computer to communicate with another. For example referring now to Figure 1, web client 100 can use HTTP to communicate with web server 150 via network 125. In this scenario the web server acts as a repository for files 160 and is capable of processing the web client's requests for such files. The files 160

stored on the web server may contain any type of data. For example, the files may contain data used to construct a form, image data, text data, or any other type of data.

HTTP has communication methods that allow web client 100 to request or send data to web server 150. The web client 100 may use web browser 110 to initiate request 115 and receive one or more files from file repository 160. Typically, web browser 110 sends a request 115 for at least one file to web server 150 and the web server forwards the requested file to web client 100 in response 120. The connection is then terminated between web client 100 and web server 150. Web client 100 then uses web browser 110 to display the requested file. A client request 115 therefore, consists of establishing a connection between the web client and the web server using network 125, issuing a response 120 to the request 115, and terminating the connection.

Web server 150 does not maintain any state information about the request once the connection is terminated. HTTP is, therefore, a stateless application protocol. That is, a web client can send several requests to a web server, but each individual request is treated independent of any other request. The web server has no recollection of any previous request. Thus, for example, if a form is completed by the user and submitted to the web server for processing, the web server does not maintain a record of the data entered into the form. One of the key reasons for the success of the HTTP approach is the fact that it is a stateless protocol. However, the stateless nature of the protocol is also what creates the current problem.

Once a file is sent from web server 150 to web client 100 it becomes ready for display. The web client's 100 web browser 110 is typically used to format and display files. Web browser 110 allows the user to request and view a file

without having to learn a complicated command syntax. Examples of several widely used web browsers include Netscape Navigator, Internet Explorer, Opera, and numerous others on PDAs, set top boxes, and screen phones. Some web browsers can display several different types of files. For example, files written using the HyperText Markup Language (HTML), the JavaScript programming language, the ActiveX programming language, or the Portable Document Format (PDF) may be displayed using a web browser. It is also possible to display various other types of files using language such as Standard Generalized Markup Language (SGML) or eXtensible Markup Language (XML).

Creating a Web Page:

A form, which provides one or more places for a user to enter data, can be embedded inside of a web page. A web page may be created using a variety of different data formats and/or programming languages. Most web pages, and as a result most forms, are created using the HyperText Markup Language (HTML). The techniques used to create a web page will now be discussed in further detail.

HTML is a language that may be used to specify the contents of a web page (e.g. web page 220). An HTML description is typically comprised of a set of markup symbols which are described in more detail below. HTML file 250 or any type of data file that contains the markup symbols for web page 220 may be sent to web browser 210. Web browser 210 executing at web client 200 parses the markup symbols in HTML file 250 and produces web page 220, which is then displayed, based on the information in HTML file 250. Web page 220 may contain text, pictures, or forms comprised of embedded text fields, checkboxes, or other types of data that is to be displayed on the web client using web browser 210. Consequently, HTML document 250 defines the web page 220 that

is rendered by web browser 210. For example, the following set of markup symbols directs web browser 210 to display a title, a heading, and an image called "image.jpg":

```
<HTML>
<HEAD>
<TITLE> This is a document title </TITLE>
</HEAD>
<BODY>
<H1> This text uses heading level one </H1>
<IMG SRC="http://www.sun.com/image.jpg">
</BODY>
</HTML>
```

In the above example, markup symbols (e.g. "<" and ">") indicate where each HTML command (e.g. TITLE) begins and ends. An HTML command, which is typically surround by markup symbols, provides the web browser with instructions to execute. Markup symbols typically surround an HTML command. The "<" symbol indicates the start of an HTML command an the "</" symbol indicates the end of an HTML command. Each start or end command has a corresponding ">" to indicate the close of that particular command. Information associated with the HTML command may be contained within the HTML command's start and end symbols. An HTML command is used to by the web browser 210 to determine how to process the block of information associated with the two commands.

In the above example, "<TITLE>", and "</TITLE>" are examples of HTML commands surrounded by markup symbols. The "</TITLE>" HTML command directs web browser 210 to place the text "This is a document title" in the title bar of web browser 210.

Some HTML commands have attribute names associated with the command. For example, HTML command "<IMG>", directs web browser 210 to display an image. A "SRC=" attribute identifies the location and name of the image to be displayed. In the above example, the statement "<IMG

SRC="http://www.sun.com/ image.jpg">" tells the web browser to display an image named "image.jpg" that can be obtained from the web server located at "http://www.sun.com."

Embedding a Form into a Web Page

An HTML file may also contain HTML commands that cause the web browser to render a web page that contains fields for entering data. The portion of the web page that contains the data entry fields may be referred to as a form. When the web page is comprised of primarily data entry fields the entire web page is sometimes also referred to as a form. As is discussed below, HTML includes an HTML form command that may cause the browser to display data entry fields. A text box, a drop down menu, a check box, a command button, a toggle button, or any other kind of interface component capable of receiving input are some examples of the type of data entry fields that may be placed in a form. Data is manually entered into the fields by the user and then submitted to a web server for processing. As previously discussed, when a user wishes to use a form to purchase a product, for example, the user manually fills in text boxes located on the form with the type of information needed to carry out the transaction. Other types of data entry fields require the user to select from one or more options such as in the case of a menu, a toggle button, or a command button, for example. When the form is complete the user submits the completed form to the vendor's web server by pressing a command button, for example.

A problem with collecting information from the user in this manner is that the user is required to manually enter data into the fields of a form. This takes time and becomes unnecessarily burdensome when the user wishes to complete more than one form with the same information.

Figure 3 provides an example of, a form created using the HTML definition language. Code block 310 contains HTML command examples. When a document comprising code block 310 is transmitted to web browser 300 executing on web client 330, it causes form 305 to be displayed. Web browser 300 displays form 305 by parsing the HTML commands contained in code block 310 and then using the information obtained to format form 305. Once the user finishes entering information into form 305, the user may submit the form by pressing command button 331. Pressing command button 331 sends the information entered by the user in form 305 to web server 320 using either the GET or the POST method.

The <FORM> command shown in code block 310 indicates the beginning of a form. Once the initial FORM command is placed into the HTML document other HTML commands may be entered between the initial FORM command and the closing FORM command (e.g. </FORM>) that represent, for example, one or more data entry fields such as a text-box, drop-down lists, check boxes, radio buttons, and input buttons. The INPUT command is used to specify different types of data entry fields. The type of data entry field created is dependant upon the value assigned to the INPUT command's TYPE attribute. For example, the INPUT command can create a text-box, a password field, a hidden field, a button, a radio button, or a checkbox. A "text" value assigned to the TYPE attribute of an INPUT command creates a text box, for example. Similarly, a TYPE of "checkbox" creates a checkbox. The following code, if inserted after the FORM element, creates a text box and a checkbox:

```
<INPUT TYPE="text" NAME="user_name">
<INPUT TYPE="checkbox" NAME="user_item1">
```

The HTML tags and text contained in code block 310, for example, create form 305 when displayed using web browser 300.

A <FORM> tag may contain an ACTION and/or METHOD attribute. The ACTION attribute specifies what program to execute when form 305 is submitted for processing. The METHOD attribute describes how data entered into the form is handled. There are two METHOD choices: GET and POST. Each one of them is capable of processing the data entered into a form. To illustrate, if a user named Duke who resides at 123 Main Street, Ca 90211 completes the Name and Address fields shown on form 305 by filling in spaces 345-347, then the following data string 350 is created and sent to web server 320.

user_name=Duke&user_address=123_Main Street,_CA_90211

This data string is handled by either the GET method or the POST method. The GET method appends information entered into form 305 by the user onto the end of a Uniform Resource Locator (URL 335) or in a QUERY_STRING environment variable. URL 335 is submitted to web server 320 by web client 330 in an HTTP request 315. Web server 320 is responsible for processing the submitted information. A problem with the GET method is that the information contained in URL 335 is not stored on web client 330 for later use and cannot therefore be later used to populate form 305. With the POST method, the information placed into form 305 by the user is placed into the data stream of the HTTP protocol. This allows web server 320 to process form 305 data using the standard input data stream. Standard input is a method for providing a computer with data that is well know to those skilled in the art. The POST method also does not provide an effective way to easily maintain and store data on web client 330 for later use without doing extra work using the Common Gateway Interface (CGI).

The ACTION attribute of the FORM tag indicates where the computer program tasked with processing data string 350 resides. For example, the following portion of code block 310 specifies that a computer program named form_processor.pl is located in cgi-bin 321 of sever 320.

```
<FORM METHOD=POST ACTION=http://server320
/form_processor.pl>
```

The form_processor.pl program is responsible for processing data string 350. Programs that utilize CGI standard are typically located in a common directory (e.g. cgi-bin 321). CGI is a specified standard for communicating between HTTP servers and server-side gateway programs. A CGI program is capable of storing data supplied by the user, but it cannot populate a form for a user located at a client computer (e.g. web client 330). CGI programs execute at the server and are therefore incapable of providing data to a form at the client.

The program "form_processor.pl" used in the above example is an example of a server-side gateway program that processes data input by the user via form 305, for example. When form 305 is submitted web server 320 executes the "form_processor.pl" program and passes the data that was entered by the user and then sent from web client 330 to server 321. That is, data string 350 is passed to "form_processor.pl" when a form is submitted. When the gateway program is done processing data string 350 the result is sent to server 320 and a response may be forwarded back to web client 330. Gateway programs can be compiled programs written in languages such as C or C++ or they can be executable scripts written using a scripting language such as PERL, TCL, or various other language. Once data string 350 is processed it may be stored on server 320 in data store 322.

Thus, existing mechanisms for processing forms reside on the server and are unable to populate a form on the client (e.g. data resides on the server and programs that process data execute on the server).

HTTP Cookies

One example of data that may be stored on the client is a cookie. A cookie is an HTTP header that consists of a text-only string. The text-string is entered into the memory of a web client and accessible by the web browser. This text-string is initially set by the computer serving the web page and consists of the domain name, path, lifetime, and value of a variable that the server sets. If the lifetime of this variable is longer than the time the user spends at that site, then the text-string is saved on the web client for later use. As a result, cookies provide a way to store and later recall values entered into a web page by the user. For example, cookies allow a web server to individually customize a web page for each user who visits the page.

A cookie may be used to populate the elements of a form that a user has previously completed. For example, referring now to Figure 9, if a user completes field 945-947 of form 905 and submits it to server 920 via submission path 915, server 920 can place the data submitted by the user in cookie 931 by sending web client 910 data via path 925. If the user returns to form 905 again Web browser 930 executing at web client 910 may use the data saved in cookie 931 populate fields 945-947 with the data previously entered by the user. Thus, a cookie can free the user from having to complete form 905 more than once. Cookies can be used to fill forms the user has previously visited. However, the data that is stored by the cookie cannot be used to populate other forms. That is, a problem with cookies is that forms located on different servers are not

provided access to the same cookie. Only web servers that are listed as having permission to access a particular cookie may obtain such access.

A cookie is only accessible to the server that set, or created it. Thus, a second form cannot use the first form's cookies to obtain the user's name and address. For example, the web server located at www.merchantA.com may be allowed to read a cookie, but the web server located at www.merchantB.com may not be allowed to read the cookie created by www.merchantA.com. This loads the disk full of redundant information. Moreover, users are not sure what type of information is stored in such cookies.

Referring now to Figure 4, the process used to create and retrieve a cookie is illustrated. Initially, web browser 401 which resides on web client 400 issues a request 410 for web page 415 from web server 450. Web server 450 responds by sending a copy of web page 415 to web client 400 via response 411. Web client 400 uses web browser 401 to display web page 415 to the user. To create a cookie 460, web server 450 sends a "Set-Cookie" command in the header line contained in response 411. For example, in response to a request 410, web server 450 may send the following command in HTTP response 411:

Set-Cookie: NAME=VALUE; expires=DATE; path=PATH; domain=DOMAIN_NAME; secure.

The NAME and VALUE parameter contain the information included in the cookie. DATE is the time at which the cookie expires and is thus no longer saved on web client 210. DOMAIN is the host address or domain name for which the cookie is valid. For example, if a cookie is set by a computer having a domain name of www.merchantA.com, only the server responsible for that particular domain name is provided access to the cookie. The PATH parameter specifies a subset of URL's at the appropriate domain for which the cookie is valid. If the keyword secure is used then the cookie is only transmitted over a secure

connection. All of these parameters except the NAME=VALUE are optional to set a cookie. Once the Set-Cookie command is sent to web client 400 a cookie 460 is placed on web client 400.

To create a cookie 460 using the UNIX operating system, for example, web server 450 could execute the following shell script in response to a request 410:

```
#!/bin/sh
echo Content-type: text/html
echo Set-cookie: FooBar=foo; expires=Wednesday,
        02-11-99 12:00:00 GMT
echo
echo <H1>A Cookie named Foobar containing the text foo is now present
on your computer</H>
```

This script stores "FooBar=foo" on web client 400. The following script allows web server 450 to read the HTTP cookie:

```
#!/bin/sh
echo Content-type: text/html
echo
echo The data supplied here was obtained from a
cookie:<P>
echo $HTTP_COOKIE
```

Once cookie 460 is created, the information stored in the cookie can be used in many different ways. Information in cookie 460 may be accessed by a script that has authorization to insert cookie 460 data into a form, for example. However, a problem with cookies is that no authentication mechanism is required to obtain the information they store. A cookie is a text file and is not stored in encrypted form. While a browser may limit access to a cookie, the text that is stored in the cookie may be accessed or read by any program that can read a text file. Therefore, it is not wise to store any sensitive data in a cookie.

A further problem is that unauthorized users may review another user's cookie files and thereby ascertain information about that user's spending, purchasing, and/or web browsing habits. Once such information is obtained it may then be utilized to target the user for theft and/or marketing.

Predefined Paste Operation

Some web browsers provide users with a mechanism for pasting predefined data into a form. Under the generic preferences menu of the Opera web browser, for example, the user can select a command button labeled personal information. When this command button is selected a dialog box that contains fields for entering personal information opens. If the user desires, the user can enter a name, an address, a phone number, an e-mail address, and/or any other kind of information into the text fields of the dialog box.

Once the user enters information into the fields of the dialog box and clicks "OK" the information that was entered can be pasted into the fields of a form by performing a right click operation. For example, if the user encounters a form that has a place for entering an address, the user can elect to paste the address previously specified in the dialog box by first selecting the address field and then right-clicking on a pointing device such as a mouse and selecting from a drop down menu the item titled "insert address".

A problem with pasting information into the fields of a form in this manner is that the user is required to manually select the place to insert the information and the type of information to insert into that place. Another problem is that unauthorized users are not prevented from accessing the information entered into the dialog box.

SUMMARY OF THE INVENTION

The present invention provides the user with a mechanism for completing a form. When a user issues a request for a form, that request is processed by a rewriting service. In one embodiment of the invention, the rewriting service provides the user with a mechanism for providing data to a form by obtaining data from a data source location. The rewriting service responds to a request for a form by generating a rewritten version of the form that contains embedded programs (e.g. applets). The applet elements identify the location of applets configured to obtain data from the data source location. A smart card is an example of a data source location. Access to the smart card may be controlled by the use of an authentication mechanism. The rewritten form is transmitted to a client computer where it utilizes the embedded programs to obtain data from a data source location such as a smart card.

The rewriting service obtains a requested form and substitutes tags of one type for tags of another type. For example, the rewriting service may substitute tags that satisfy a set of predetermined criteria for tags that identify the location of an executable program. In one embodiment of the invention, this is accomplished by replacing XML elements (e.g. tags) that are inserted into the requested form in a standardized format with APPLET elements that identify the location of an executable applet program. This enables browsers that are not capable of parsing XML to utilize the advantages of the present invention. The rewriting service can replace elements of one type with elements of a second type. For example, any of the elements contained in a form (e.g. an HTML document) may be replaced with APPLET elements. FORM tags or INPUT tags, for example, may also be replaced with APPLET elements.

The APPLET elements identify the location of an applet program configured to obtain data from a data source. A smart card or other type of data store is an example of a data source.

One embodiment of the invention contemplates the user of an authentication mechanism to provide the user with control over the type of information utilized to complete a form. When the applet initially seeks to obtain data from a data source it may prompt the user to insert a smart card. When the user inserts the smart card the user may be prompted for authentication information. If the user enters the correct information, the applet is allowed to read data from the smart card. If the user does not enter the correct information then access to the data stored on the smart card is not allowed.

One embodiment of the invention comprises computer code configured to advertise services offered by the proxy, the rewriting service, the applets, the server computer, the card server, or the client computer. A service is an entity that can be used by a user, a program, or another service to perform a certain kind of predetermined functionality. The ability to dynamically advertise services provides a mechanism to personalize the tasks performed by a service. The user, for example, could customize the way a particular type of form is to be completed. A user that wants to be prompted before the system disclose his social security number could elect to have this event occur whenever a form is encountered that prompts the user for a social security number.

DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of the various components used by the World Wide Web.

Figure 2 is a block diagram illustrating the components used to create a web page.

Figure 3 is a block diagram illustrating how web client interacts with a web server to display a form.

Figure 4 illustrates the process used to create and retrieve a cookie.

Figure 5 is an illustration of the components used by one embodiment of the invention to provide the user with a mechanism for completing a form.

Figure 6 is a flow chart illustrating the steps performed by one embodiment of invention.

Figure 7 is a block diagram illustrating the input transmitted to the rewriting service and the output generated by the rewriting service in one embodiment of the invention.

Figure 8 is an example of a form used by one embodiment of the invention to collect data.

DETAILED DESCRIPTION

A method and apparatus for completing a form is described. In the following description numerous specific details are set forth in order to provide a more thorough understanding of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details. In other instances, well-known features have not been described in detail so as not to obscure the invention.

The present invention provides the user with a mechanism for completing a form. In one embodiment of the invention, a rewriting service, executing at the server, responds to a request for a form by generating a version of the form that contains embedded programs (e.g. applets). The rewritten form is transmitted to a client computer where it obtains data from a data source location such as a smart card.

The rewriting service provides the user with a mechanism for providing data to a form by obtaining data from a data source location. In one embodiment of the invention, applet elements are inserted by the rewriting service. The applet elements identify the location of applets configured to obtain data from the data source location. A smart card is an example of a data source location. Access to the smart card may be controlled by the use of an authentication mechanism.

Generating a Form:

A form is a type of digital document that contains one or more data receptacles. It is possible to create a form using a variety of different computer programming languages. The invention contemplates the use of forms created

using HyperText Markup Language (HTML), eXtensible Markup Language (XML), Standardized General Markup Language (SGML), JavaScript, the Java programming language, or any other programming language capable of generating a document that provides a location for the user to enter data (e.g. a data receptacle). For example, a spreadsheet, a word processing document, , a database table, or any other type of document having one or more data receptacles may be referred to as a form. It is also possible to create a form using high-level programming languages such as C/C++, Pascal, or Smalltalk. Forms are typically generated by a computer program and presented to the user via a Graphical User Interface (GUI). For example, a web browser, an operating system, or any other type of computer program may be utilized to display a form to the user. In one embodiment of the invention, any type of computer program capable of executing a layout engine can generate a form. For example, the NGLayout engine which is based on open Internet standards such as HTML 4.0, CSS 1/2, XML 1.0, and the Document Object Model can be embedded into various execution environments and used to generate a form. A form may be pre-created and stored on a server computer until a request for the form is issued. However, it is also feasible to store the form locally. For example, a form may reside on a client computer instead of the server computer. In either case, the form is delivered to the user when a request is issued. In accordance with one embodiment of the invention, a form is not generated until it is requested by the user. This is referred to as generating a form dynamically or "on the fly" To generate a form in such a manner the server contains computer program code configured to generate a form. When the user initiates a request for a form, the server executes the computer program code and thereby generates a form. The form is then transmitted to the client for display.

Data Receptacles:

In one embodiment of the invention, each form contains one or more data receptacles. Data receptacles are designed to obtain and/or receive information. Information may be placed inside of a data receptacle by the user or by a computer program configured to perform an insert operation. For example, it is possible to populate a data receptacle with text or numerical information. Data receptacles may also be referred to as fields, form elements, or text boxes.

Referring now to Figure 8, an example of a form used to collect data is shown. Form 800 may be displayed to the user in a web browser 801. Form 800 utilizes data receptacles 804-812 to collect information about a user. In one embodiment of the invention, form 800 is embedded inside of a web browser 801. Data receptacles 804-807 may be used to collect billing information such as the name, company, address, phone number, and e-mail address of the party responsible for payment. Data receptacles 808-811 may be used to collect shipping information when the shipping information is different than the billing information. Data receptacle 812 is intended to collect credit card or other payment information. One embodiment of the invention provides a way to populate data receptacles with data without requiring the user to manually enter the data.

Overview of the Form Completion Process:

Figure 5 is an illustration of the components used by one embodiment of the invention to provide the user with a mechanism for completing a form. A request 503 for form 505 is typically initiated at client computer 500 by a computer program capable of displaying form 505 to the user. A web browser 507, for example, may transmit request 503 to server 509 via interconnection fabric 511. When server 509 receives request 503, it may be handed to proxy 515

for processing. In one embodiment of the invention, proxy 515 executes rewriting service 517. However, request 503 may also be handed directly to rewriting service 517 and the rewriting service may execute without proxy 515. Rewriting service 517 executes a resynchronization process that obtains form 505 from data store 590 and modifies it. Thus, form 505 becomes rewritten as form 506.

The modified version of form 505 (e.g. form 506) contains embedded code configured to interact with card server 519. The rewriting service, for example, may replace existing HTML FORM elements with APPLET elements and thereby provide a way for client computer 500 to obtain applets 525-528 from server computer 509. When rewriting service 517 generates form 506, server 509 may transmit form 506 to client 500 via response 556. Response 556 may be sent across interconnection fabric 511 to client 500. Once the rewritten form resides on client 500, the applets identified in the APPLET elements may be executed. In one embodiment of the invention, applets 525-528 are obtained from data store 590 located on server 509. However, applets 525-528 may be obtained from any source connected to interconnection fabric 511.

Applets 525-528 execute in web browser 507 and communicate with card server 519 to obtain data 521 from a data source location. For example, card server 519 may be configured to read data 521 from smart card 523. Data 521, however, may also reside on other locations, such as server 509. In one embodiment of the invention, data 521 is obtained from smart card 523 and sent to web browser 507 for display in data receptacles 529-532. Data receptacles 529-532 and the information placed in them become visible to the user when form 506 is shown on display 550.

Figure 6 is a flow chart illustrating one embodiment of invention. At step 600, a request for a form is issued. A request is typically initiated by the user of a client computer, but may also be initiated by computer code, executing at the client, configured to automatically request a form when an event occurs. For instance, a form prompting the user for registration info (e.g. name, address, etc...) may be automatically presented to the user when the user installs a new computer program. The invention also contemplates the use of a web browser to request a form. The entity that initially issues the request is referred to as a requestor.

In one embodiment of the invention, the requested form contains XML tags embedded into it. However, the form may also contain data written in other programming languages such as HTML, SGML, or any other language compatible with the protocols utilized to transmit data across a computer network. The requested form resides on a source computer. The source computer may be a server computer, a client computer, or any other computer accessible via a computer network. In one embodiment of the invention, the requested form is processed by a rewriting service before it is forwarded to the requestor (e.g. the web browser). The rewriting service typically resides on the server computer, but may be placed on any computer capable of functioning as a source computer. Once a form is requested, step 603 executes and the source computer obtains the requested form from the source computer.

At step 605, the requested form is forwarded to a rewriting service. In one embodiment of the invention, the requested form is transmitted across a computer network to the computer containing the source computer. However, If the rewriting service resides on the source computer, the requested form can be forwarded directly to the rewriting service without having to be transmitted across a computer network. Once the rewriting service receives the requested

form it executes step 607 where it processes the form and thereby prepares the form for transmission to the requestor. An additional possibility is that the supplied form can be signed using the rewriting service. At step 609, the rewritten form is transmitted to the requestor for display. For example, the rewritten form may be sent to a web browser which parses the information comprising the form to generate an image for display to the user. If, for example, the rewriting service resides on the server computer, the rewritten form is transmitted across a computer network to the client computer for display. If, the rewriting service does not reside on the server computer, but instead resides on the client, then the rewritten form is sent directly to the program responsible for displaying the form to the user (e.g. the web browser).

Displaying the rewritten form causes step 611 to execute. At step 611, the APPLET elements encompassed into the rewritten form are parsed. In one embodiment of the invention, this causes the client computer to obtain and execute the applets referred to in the APPLET element. For example, if one of the APPLET elements contained in the rewritten form refer to an applet named ValidateField, the ValidateField applet will execute. The invention contemplates the use of applets configured to obtain data from a data source (e.g. at step 613). For example, in one embodiment of the invention, the applets acquire data from a smart card server configured to communicate with a smart card that is inserted into the client computer by the user. However, the applets may be configured to obtain data from other types of data sources made available to the client computer by a computer network. In one embodiment of the invention, access to the data source is not permitted until the user is authenticated. For example, the user may be required to enter a user name and password before data is retrieved from the smart card. Other authentication mechanisms, such as encryption may also be integrated into the authentication process. At step 615,

the invention places the data accumulated from the data source location into one or more of the data receptacles present on the form initially requested by the requestor and it posts the form back to the server.

The Rewriting Service:

When a user issues a request for a form, that request is processed by a rewriting service. The rewriting service obtains a requested form and substitutes tags of one type for tags of another type. For example, the rewriting service may substitute tags that satisfy a set of predetermined criteria for tags that identify the location of an executable program. In one embodiment of the invention, this is accomplished by replacing XML elements (e.g. tags) that are inserted into the requested form in a standardized format with APPLET elements that identify the location of an executable applet program. The rewriting service can replace elements of one type with elements of a second type. For example, any of the elements contained in a form (e.g. an HTML document) may be replaced with APPLET elements. FORM tags or INPUT tags, for example, may also be replaced with APPLET elements.

For example, referring now to Figure 7, if code block 700 is processed by rewriting service 705, code block 707 is generated. Line 701, for example, becomes line 708. Thus, <FORM> is translated into <form name = form_name> once it is processed by the rewriting service. The INPUT tags, shown at lines 702 and 703 become parameters for an applet. The NAME parameter associated with the INPUT tag, for example, is inserted as a parameter for an applet named ValidateField. As a result, line 702 which reads <INPUT TYPE="text" NAME="something_date value"> becomes lines 709-713 which reads

```
<APPLET NAME="ValidateField">
    <PARAM NAME=type VALUE=data>
    <PARAM NAME=something_date value=name>
    <PARAM NAME=init_value value="">
</APPLET>.
```

The NAME parameter in line 702 may also be inserted into code block 707 as a hidden field. Line 714 is an example of a hidden field. When an INPUT value is passed a type parameter of hidden, nothing is displayed to the user. However, the hidden value is accessible by a computer program configured to process the form. Not all lines processed by the rewriting service are rewritten. Line 703, for example, remains unaltered by the rewriting service. Thus, line 703 is identical to line 716. The rewriting service may also partially alter lines in the originally requested form. In one embodiment of the invention, the rewriting service is configured to insert additional lines. Therefore, the rewritten form may contain more or less lines than originally in the requested form. Once the rewriting service completes the rewriting process, the form that is generated is transmitted to the requestor (e.g. the client computer) for display. At this point, the applets referred to in the APPLET elements may execute.

Extensible Markup Language (XML) is a technology that provides a way to separate the value of the data from the presentation of the data. The present invention also works with XML. Using XML the HTML portion of the form can be populated with XML tags.

The Applet:

An applet is a small computer program configured to perform one or more predefined tasks. In one embodiment of the invention, applets obtain data from a data source location and insert that data into one or more data receptacles on behalf of a user. For example, once the rewriting service has properly embedded information into the web page, the applet can populate a

data receptacle with the information needed to complete a commercial transaction. An applet may obtain data from a variety of different sources. Any type of static, dynamic, random access, or read only memory device may be programmed to retain the type of data utilized by the rewriting service.

A smart card or flash memory device, for example, can store such data. Other memory devices such as a hard drive, a CD-ROM, a CD-R, a DAT, or a DVD device can also retain data for use by the applet. The applet may access the memory device locally or over a communication medium such as a computer network. In one embodiment of the invention, the memory devices contain personal information about a user. However, it is feasible for the applet to process any kind of data. The applet, for example, may also process business data and/or credit card data.

Authentication:

The applets may be configured to obtain data from a smart card (e.g. the data source location). When the user inserts a smart card the smart card communicates with a card server. In one embodiment of the invention, the card server resides on the user's system. However, the card server may be placed on any system accessible via a computer network. Access card is not permitted unless the user inserting the smart card is authenticated. The present invention contemplates the use of various authentication mechanisms. For example, the user may be required to enter a user name and password before the card server is permitted to read data from the smart card. Various other techniques for ensuring the data on the smart card is only provided authorized sources may also be employed. A PIN number or similar method of authenticating the user, for example, may also be utilized to ensure that only authorized access to the

smart card occurs. Biometric instruments may also be utilized to determine if a person is authorized to access the smart card.

The authentication mechanism provides the user with control over the type of information utilized to complete a form. When the applet initially seeks to obtain data from a data source it may prompt the user to insert a smart card. When the user inserts the smart card the user may be prompted for authentication information. If the user enters the correct information, the applet is allowed to read data from the smart card. If the user does not enter the correct information then access to the data stored on the smart card is not allowed.

Advertising Services:

One embodiment of the invention comprises computer code configured to advertise the services offered by the proxy, the rewriting service, the applets, the server computer, the card server, or the client computer. A service is an entity that can be used by a user, a program, or another service to perform a certain kind of predetermined functionality. The Jini ™ architecture provides a mechanism for integrating services into a single, dynamic distributed system. Services, for example, are capable of providing computational functionality, storage, a communication channel to another user, a software filter, access to a hardware device, or another user.

Members of a Jini ™ system federate in order to share access to services. For example, if the card server joins the federation it could provide the functionality it offers to another service. Members of a Jini ™ system are capable of utilizing the services each member offers to collectively perform a particular task. For example, services may make use of another service, and a client of one service may itself be a service for clients of its own.

The rewriting service, for example, may be configured to interact with the card server to perform a task for the user. In one embodiment of the invention, the ability to dynamically advertise services provides a mechanism to personalize the tasks performed by a service. The user, for example, could customize the way a particular type of form is to be completed. A user that wants to be prompted before the system discloses his social security number could elect to have this event occur whenever a form is encountered that prompts the user for a social security number.

Thus, a method and apparatus for completing a form is described.

CLAIMS

1. A method for completing a form comprising:
initiating a request for a form having at least one data receptacle;
obtaining said form in response to said request;
providing said form to a rewriting service.

2. The method of claim 1 wherein said rewriting service generates a rewritten form by substituting a first type element in said form for a second type element.

3. The method of claim 2 wherein said second type element identifies the location of program code.

4. The method of claim 2 further comprising:
parsing said rewritten form;
displaying said rewritten form to a user.

5. The method of claim 4 further comprising:
determining whether said user is authenticated;
obtaining data from a data source location when said user is authenticated;
providing said data to said at least one data receptacle.

6. The method of claim 5 wherein said data source location is smart card.

7. The method of claim 6 wherein a card server interacts with said smart card.

8. The method of claim 1 wherein said rewriting service executes on a server.

9. The method of claim 3 wherein said program code is an applet.

10. The method of claim 5 wherein said rewriting service offers services that allow said user to customize when said step of providing data to said at least one data receptacle occurs.

11. The method of claim 7 wherein said card server offers services to said rewriting service.

12. A computer program product comprising:

a computer usable medium having computer readable program code embodied therein for completing a form, said computer program product comprising:

computer readable program code configured to obtain a form having data receptacles in response to a request;

computer readable program code configured to generate a rewritten form by substituting a first type element in said form for a second type element.

13. The computer program product of claim 12 wherein said second type element identifies the location of program code.

14. The computer program product of claim 12 further comprising:

computer readable program code configured to provide said rewritten form to a requestor.

computer readable program code configured to parse said rewritten form at said requestor;

computer readable program code configured to display said rewritten form to a user at said requestor.

15. The computer program product of claim 14 further comprising:

computer readable program code configured to determine whether said user is authenticated;

computer readable program code configured to obtain data from a data source location when said user is authenticated;

computer readable program code configured to provide said data to said at least one data receptacle.

16. The computer program product of claim 15 wherein said data source location is smart card.

17. The computer program product of claim 16 wherein a card server interacts with said smart card.

18. The computer program product of claim 13 wherein said program code is an applet.

19. The computer program product of claim 15 further comprising:
computer readable program code configured to offer services that allow said user to customize when said computer readable program code provides said data to said at least one data receptacle.

20. A method for completing a form comprising:
initiating a request for a form having at least one data receptacle;
obtaining said form in response to said request;
providing said form to a rewriting service;
executing said rewriting service to generate a rewritten form by substituting a first type element in said form for a second type element;

21. The method of claim 20 wherein said second type element identifies the location of program code.

22. The method of claim 20 further comprising:
parsing said rewritten form;
displaying said rewritten form to a user;

23. The method of claim 22 further comprising:
determining if said user is authenticated;
obtaining data from a data source location when said user is authenticated;
providing said data to said at least one data receptacle.

24. The method of claim 23 wherein said data source location is smart card.

25. The method of claim 24 wherein a card server interacts with said smart card.

26. The method of claim 20 wherein said rewriting service executes on a server.

27. The method of claim 21 wherein said program code is an applet.

28. The method of claim 23 wherein said rewriting service offers services that allow said user to customize when said step of providing data to said at least one data receptacle occurs.

29. A method for completing a form comprising:

initiating a request for a form having at least one data receptacle;

obtaining a rewritten form having program code, said program code configured to communicate with a data source location having data;

parsing said rewritten form;

executing said program code to obtain said data from said data source location;

displaying said rewritten form containing said data to a user;

30. The method of claim 29 wherein said rewriting service generates said rewritten form by substituting a first type element in said form for a second type element.

31. The method of claim 30 wherein said second type element identifies the location of said program code.

32. A method for completing a form comprising:

responding to a request for a form by obtaining a form;

providing said form to a rewriting service, said rewriting service generating a rewritten form by substituting a first type element in said form for a second type element;

transmitting said rewritten form to a client computer for processing.

33. The method of claim 32 wherein said second type element identifies the location of said program code, said program code configured to obtain data from a data source connected to said client computer.

34. A method for completing a form comprising:

a means for initiating a request for a form having at least one data receptacle;

a means for obtaining said form in response to said request;

a means for providing said form to a rewriting service;

a means for executing said rewriting service to generate a rewritten form by substituting a first type element in said form for a second type element.

35. The method of claim 34 wherein said second type element identifies the location of program code.

36. The method of claim 34 further comprising:

a means for parsing said rewritten form;

a means for displaying said rewritten form to a user.

37. The method of claim 36 further comprising:

a means for determining if said user is authenticated;

a means for obtaining data from a data source location when said user is authenticated;

a means for providing said data to said at least one data receptacle.

38. The method of claim 34 wherein said data source location is smart card.

39. The method of claim 34 wherein a card server interacts with said smart card.

40. The method of claim 34 wherein said rewriting service executes on a server.

41. The method of claim 35 wherein said program code is an applet.

42. The method of claim 37 wherein said rewriting service offers services that allow said user to customize when said step of providing data to said at least one data receptacle occurs.

FIGURE 1

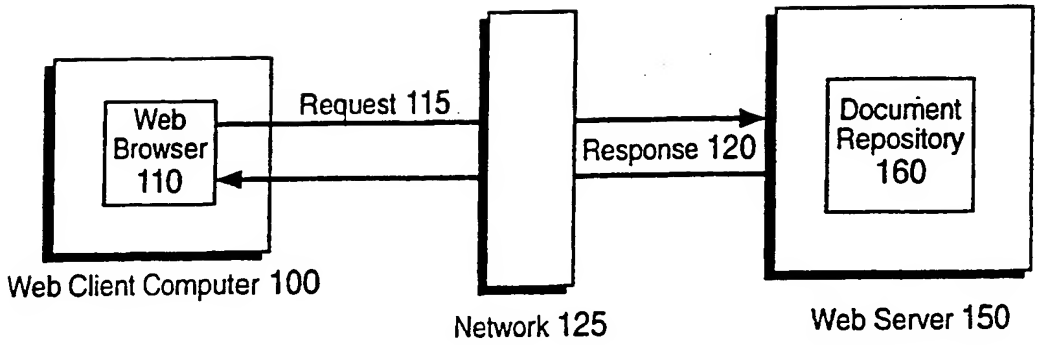
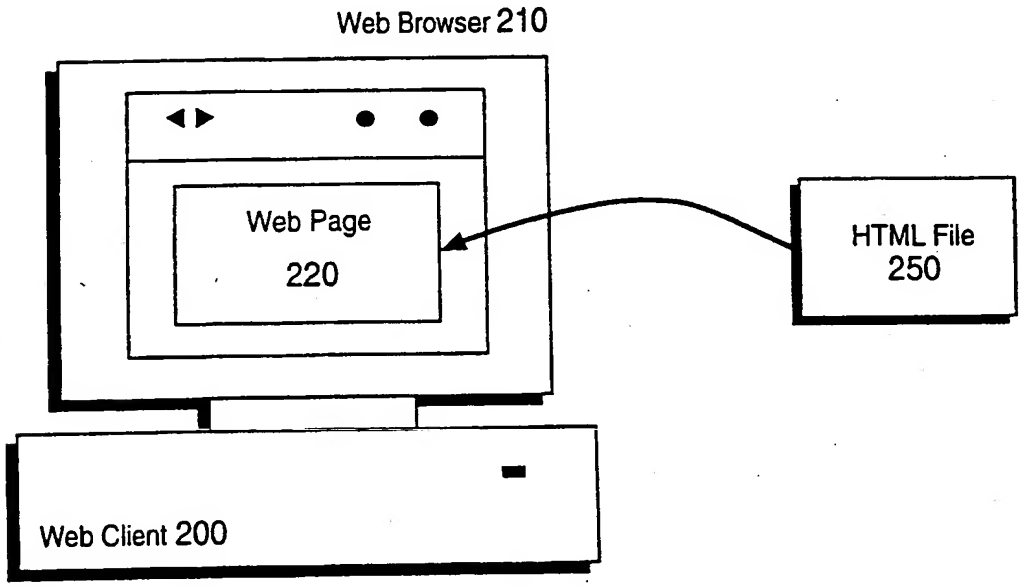


FIGURE 2



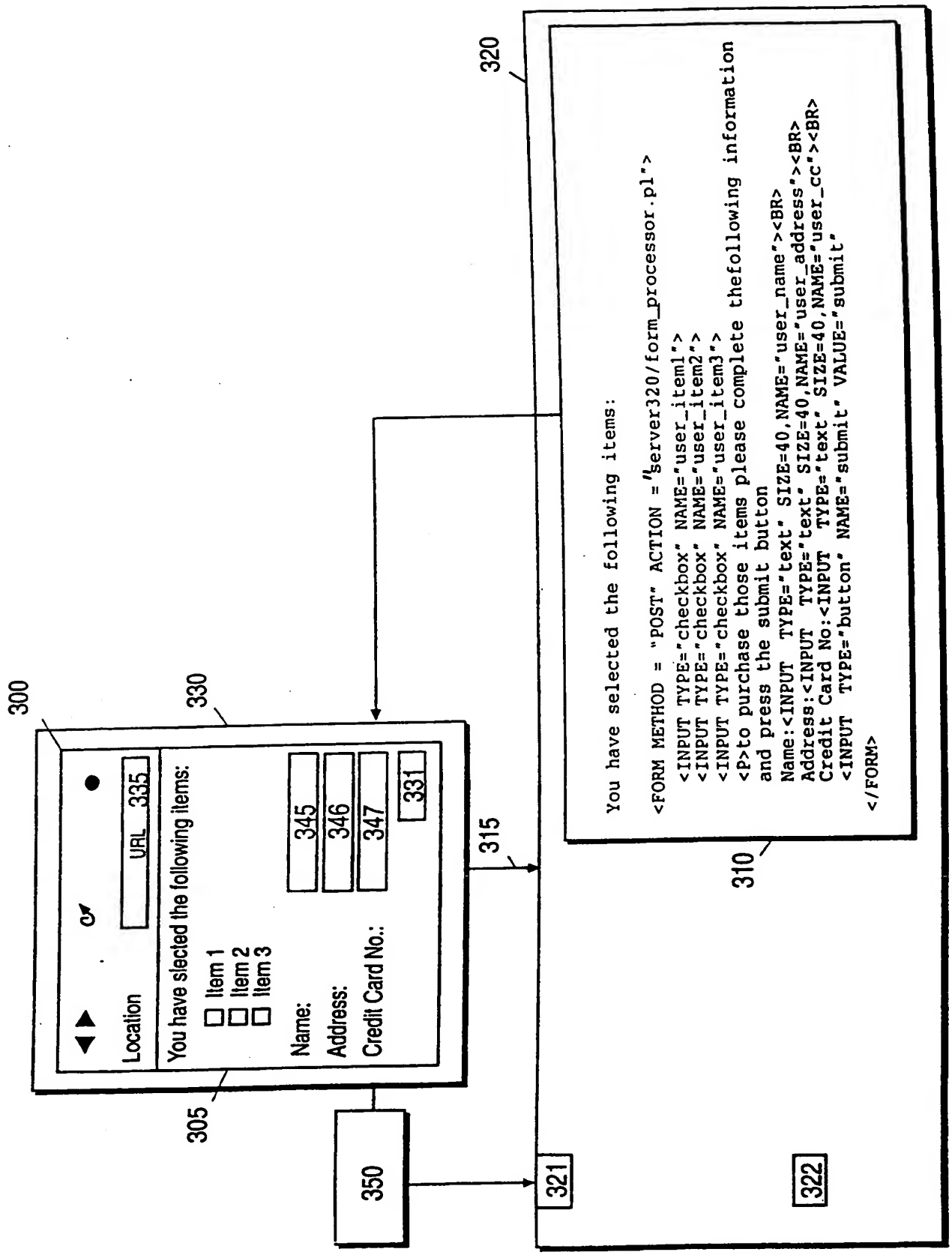

300
330
320
Location
URL 335
You have slected the following items:
Item 1
Item 2
Item 3
Name:
Address:
Credit Card No.:
345
346
347
331
315
305
350
321
322
310
You have selected the following items:
<FORM METHOD = "POST" ACTION ="server320/form_processor.pl">
<INPUT TYPE="checkbox" NAME="user_item1">
<INPUT TYPE="checkbox" NAME="user_item2">
<INPUT TYPE="checkbox" NAME="user_item3">
<P>to purchase those items please complete thefollowing information
and press the submit button
Name:<INPUT TYPE="text" SIZE=40,NAME="user_name">
Address:<INPUT TYPE="text" SIZE=40,NAME="user_address">
Credit Card No:<INPUT TYPE="text" SIZE=40,NAME="user_cc">
<INPUT TYPE="button" NAME="submit" VALUE="submit"
</FORM>

FIGURE 3

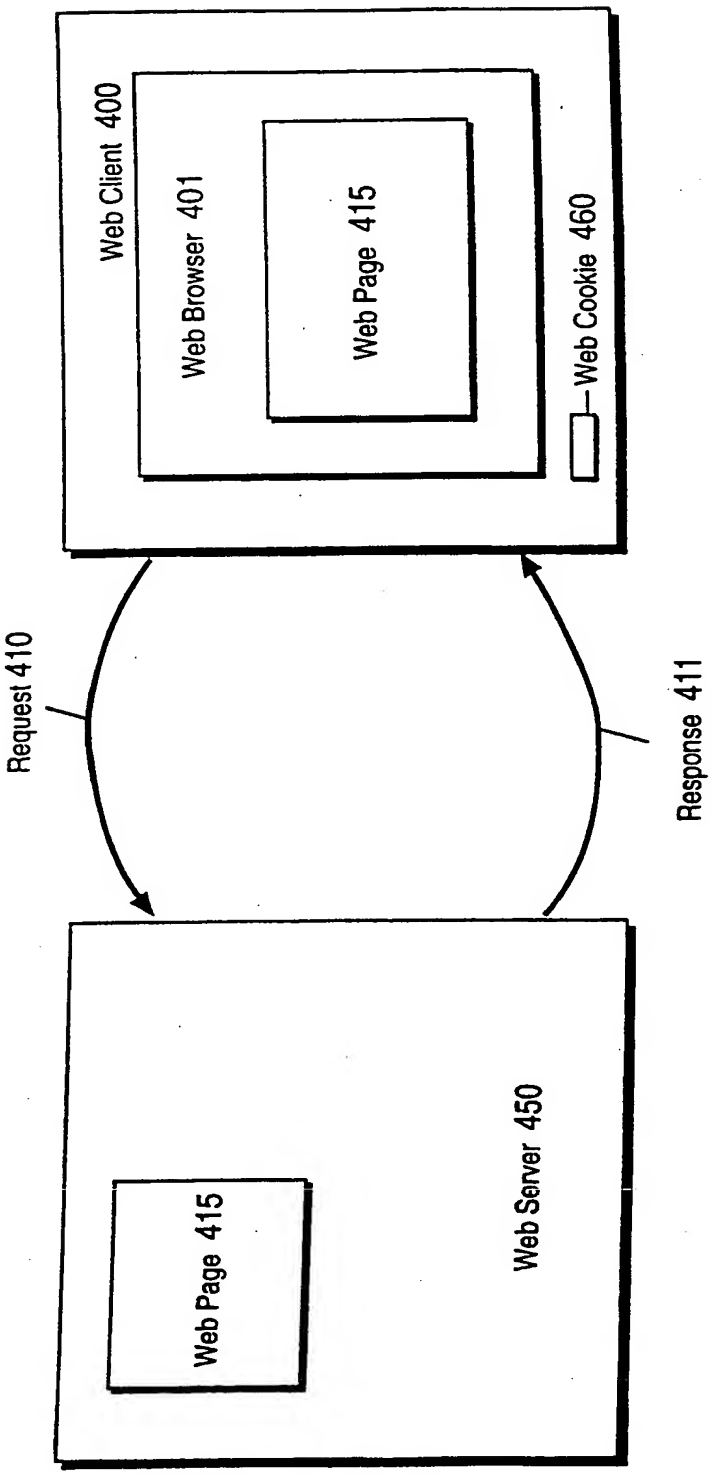

Web Client 400
Web Browser 401
Web Page 415
Web Cookie 460
Request 410
Response 411
Web Server 450
Web Page 415

FIGURE 4

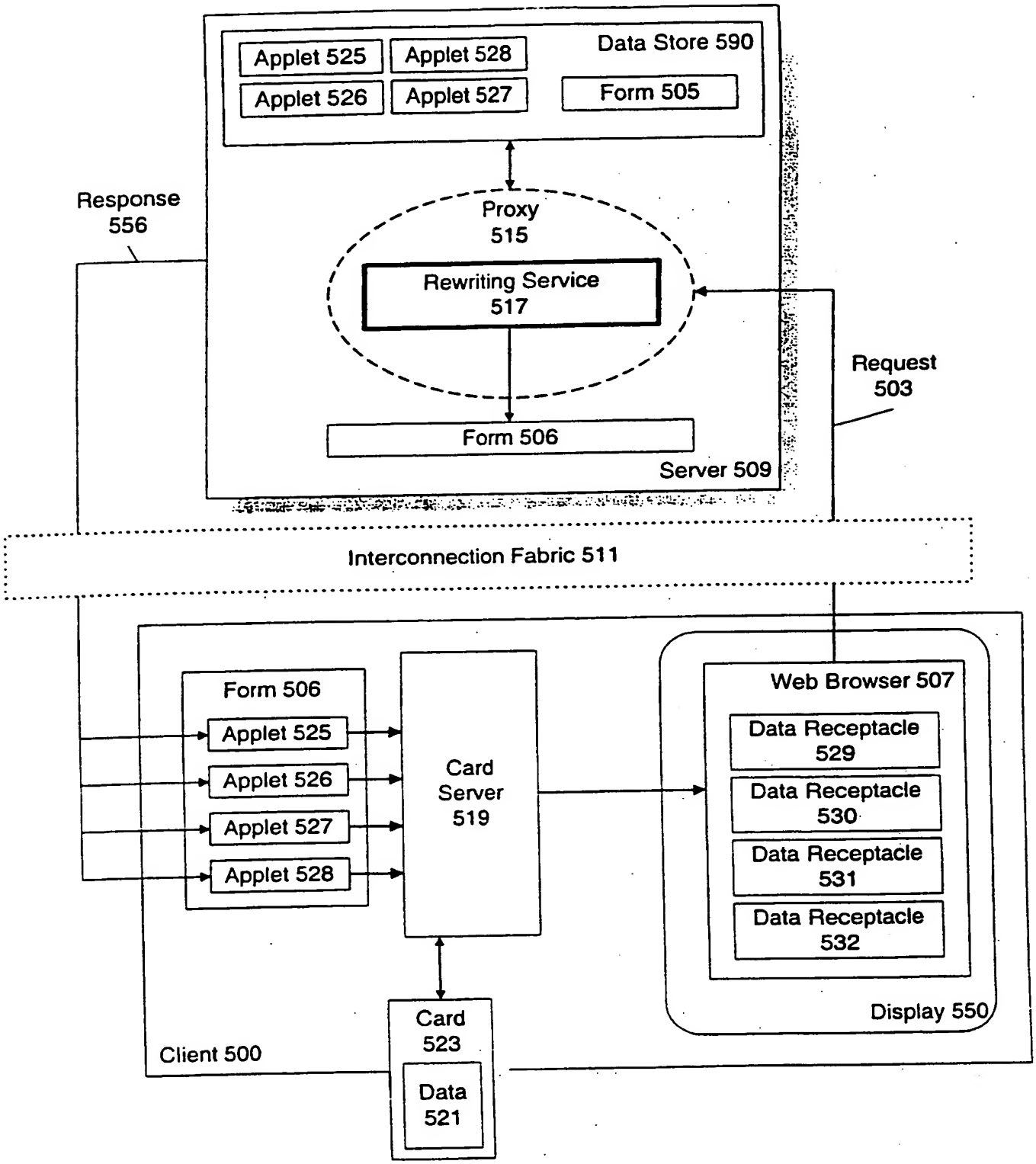

Applet 525
Applet 528
Applet 526
Applet 527
Data Store 590
Form 505
Response
556
Proxy
515
Rewriting Service
517
Request
503
Form 506
Server 509
Interconnection Fabric 511
Form 506
Applet 525
Applet 526
Applet 527
Applet 528
Card
Server
519
Web Browser 507
Data Receptacle
529
Data Receptacle
530
Data Receptacle
531
Data Receptacle
532
Display 550
Card
523
Data
521
Client 500

FIGURE 5

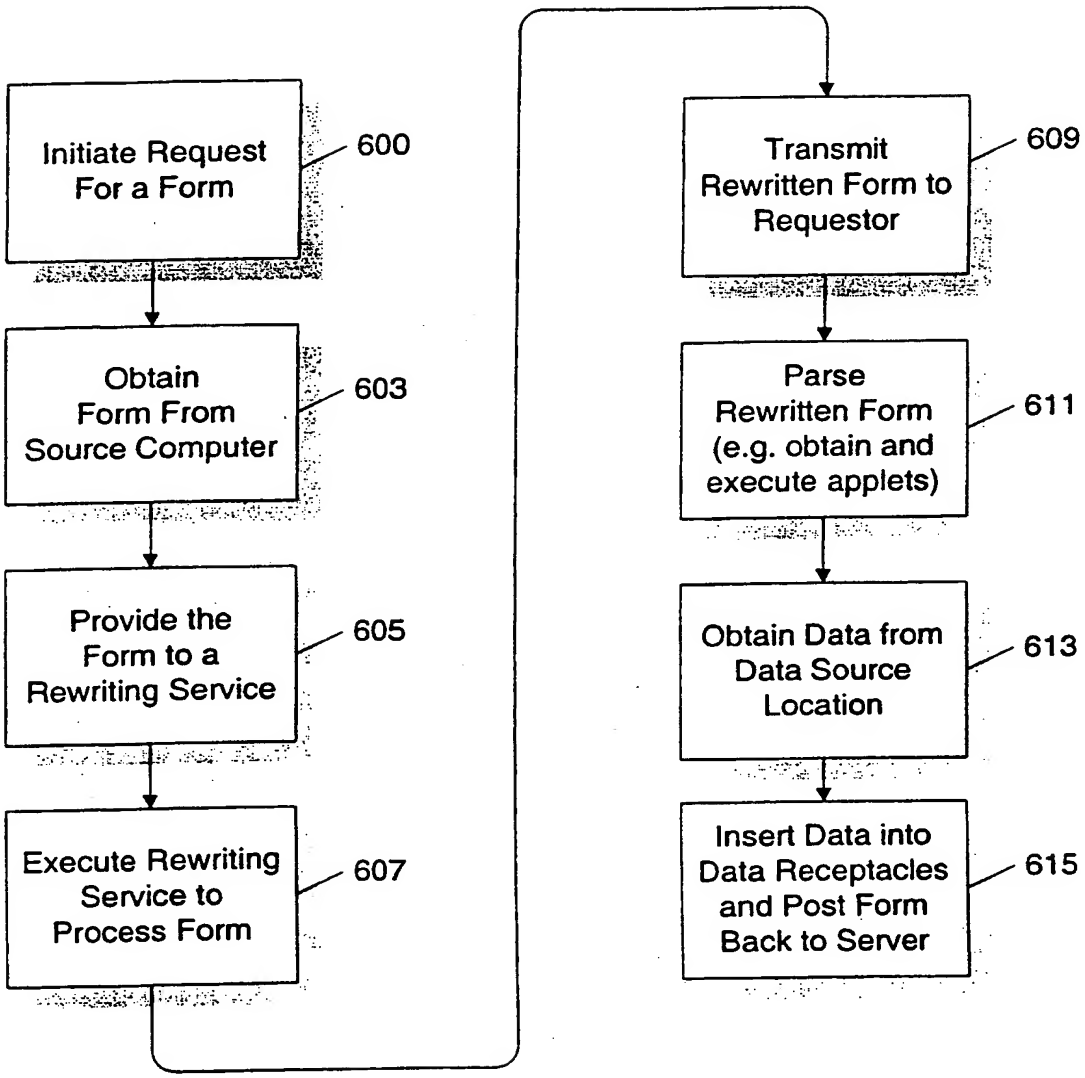

Initiate Request For a Form
600
Obtain Form From Source Computer
603
Provide the Form to a Rewriting Service
605
Execute Rewriting Service to Process Form
607
Transmit Rewritten Form to Requestor
609
Parse Rewritten Form (e.g. obtain and execute applets)
611
Obtain Data from Data Source Location
613
Insert Data into Data Receptacles and Post Form Back to Server
615

FIGURE 6

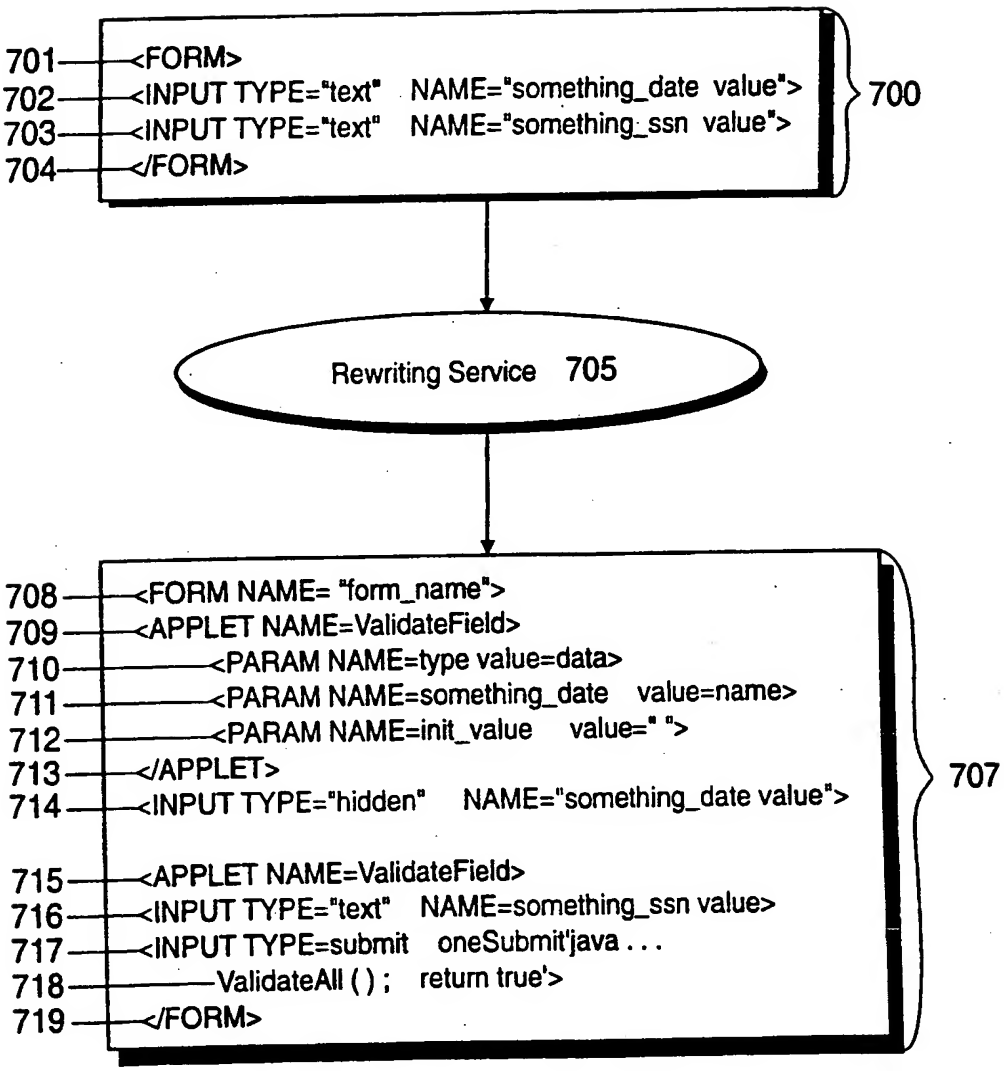


FIG. 7

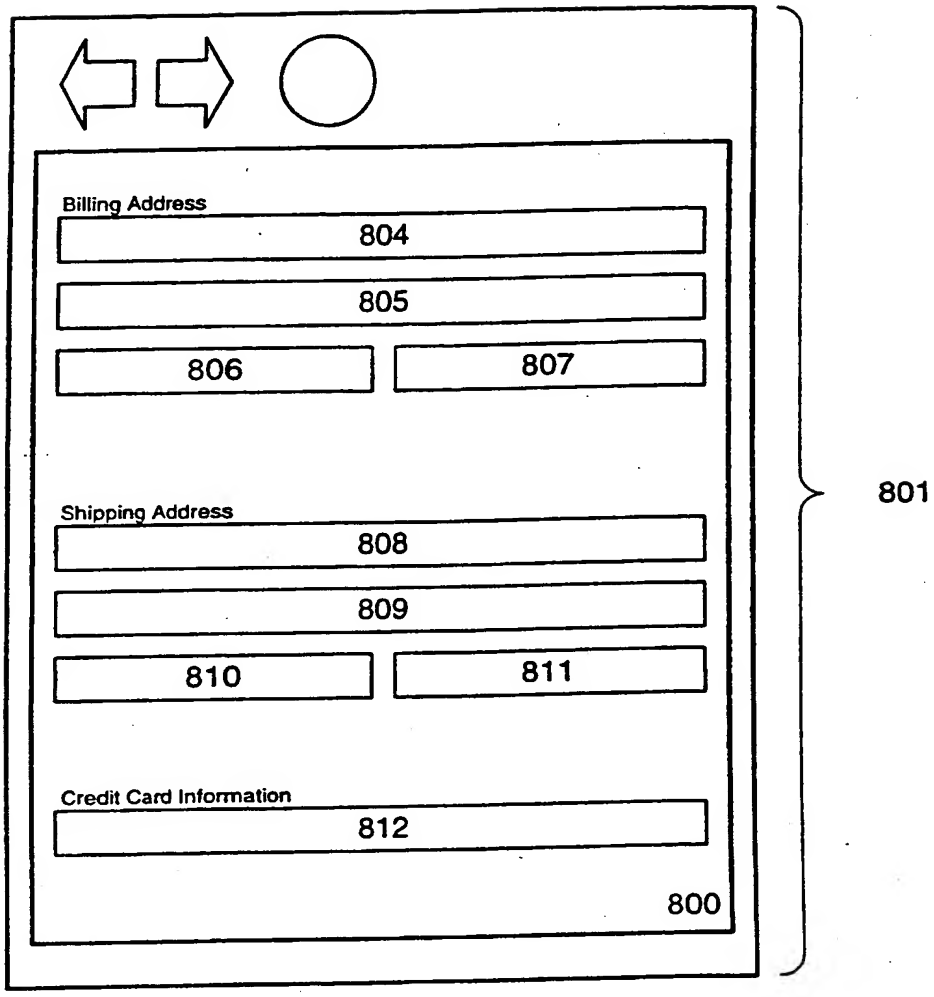


FIGURE 8

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 April 2001 (12.04.2001)

PCT

(10) International Publication Number
WO 01/025873 A3

(51) International Patent Classification[7]: G06F 17/24

(21) International Application Number: PCT/US00/27191

(22) International Filing Date: 3 October 2000 (03.10.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/414,402 7 October 1999 (07.10.1999) US

(71) Applicant: SUN MICROSYSTEMS, INC. [US/US]; 901 San Antonio Road, M/S: UPAL01-521, Palo Alto, CA 94303 (US).

(72) Inventors: DIGIORGIO, Rinaldo; 20 Mile Common Road, Easton, CT 06612 (US). UHLER, Stephen; Mundell Way, Los Altos, CA 94022 (US).

(74) Agents: MCKAY, Philip, J. et al.; Gunnison, McKay & Hodgson, L.L.P., 1900 Garden Road, Suite 220, Monterey, CA 93940 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

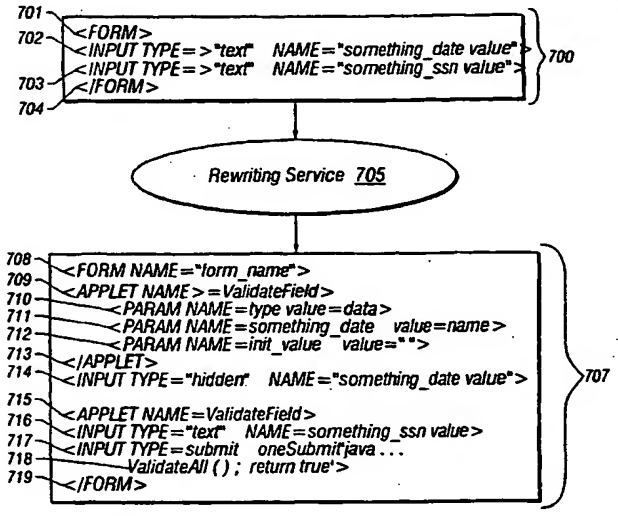
(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW). Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM). European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(88) Date of publication of the international search report:
11 July 2002

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD AND APPARATUS FOR COMPLETING A FORM



(57) Abstract: The present invention provides the user with a mechanism for completing a form. When a user issues a request for a form, that request is processed by a rewriting service. In one embodiment of the invention, the rewriting service provides the user with a mechanism for providing data to a form by obtaining data from a data source location. The rewriting service responds to a request for a form by generating a version of the form that contains embedded programs (e.g. applets). The applet elements identify the location of applets configured to obtain data from the data source location. A smart card is an example of a data source location. Access to the smart card may be controlled by the use of an authentication mechanism. The rewritten form is transmitted to a client computer where it utilizes the embedded programs to obtain data from a data source location such as a smart card.

WO 01/025873 A3

INTERNATIONAL SEARCH REPORT

Interr onal Application No
PCT/US 00/27191

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F17/24

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	ZHAO Y: "WEBENTREE: A WEB SERVICE AGGREGATOR" IBM SYSTEMS JOURNAL, IBM CORP. ARMONK, NEW YORK, US, vol. 37, no. 4, 1998, pages 584-595, XP001002913 ISSN: 0018-8670 page 589, column 1 -page 590, column 2 page 592 page 594, last paragraph --- -/--	1-42

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
4 April 2002	11/04/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Triest, J

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inter ɔnal Application No
PCT/US 00/27191

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	JACOBS S ET AL: "Filling HTML forms simultaneously: CoWeb - architecture and functionality" COMPUTER NETWORKS AND ISDN SYSTEMS, NORTH HOLLAND PUBLISHING. AMSTERDAM, NL, vol. 28, no. 11, 1 May 1996 (1996-05-01), pages 1385-1395, XP004018236 ISSN: 0169-7552 abstract page 1387 ---	1-4, 6-14, 16-22, 24-27, 29-32, 34-36, 38-41
A	US 5 794 259 A (KIKINIS DAN) 11 August 1998 (1998-08-11) abstract ---	1-42
A	GABBER E ET AL: "HOW TO MAKE PERSONALIZED WEB BROWSING SIMPLE, SECURE, AND ANONYMOUS" FINANCIAL CRYPTOGRAPHY. INTERNATIONAL CONFERENCE, XX, XX, 1997, pages 17-31, XP001011338 abstract page 21 -page 24 ---	1-42
A	GUTHERY: "JAVA CARD: Internet Computing on a Smart Card" IEEE INTERNET COMPUTING, IEEE SERVICE CENTER, PISCATAWAY, NJ, US, 1 February 1997 (1997-02-01), pages 57-59, XP002077647 ISSN: 1089-7801 the whole document -----	6,7,11, 16,17, 24,25, 38,39

1

INTERNATIONAL SEARCH REPORT

...ormation on patent family members

Inter ...nal Application No
PCT/US 00/27191

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5794259 A	11-08-1998	WO 9804976 A1	05-02-1998

Form PCT/ISA/210 (patent family annex) (July 1992)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

CORRECTED VERSION

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 April 2001 (12.04.2001)

PCT

(10) International Publication Number
WO 01/025873 A3

(51) International Patent Classification[7]: G06F 17/24

(21) International Application Number: PCT/US00/27191

(22) International Filing Date: 3 October 2000 (03.10.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/414,402 7 October 1999 (07.10.1999) US

(71) Applicant: SUN MICROSYSTEMS, INC. [US/US]; 901 San Antonio Road, M/S: UPAL01-521, Palo Alto, CA 94303 (US).

(72) Inventors: DIGIORGIO, Rinaldo; 20 Mile Common Road, Easton, CT 06612 (US). UHLER, Stephen; Mundell Way, Los Altos, CA 94022 (US).

(74) Agents: MCKAY, Philip, J. et al.; Gunnison, McKay & Hodgson, L.L.P., 1900 Garden Road, Suite 220, Monterey, CA 93940 (US).

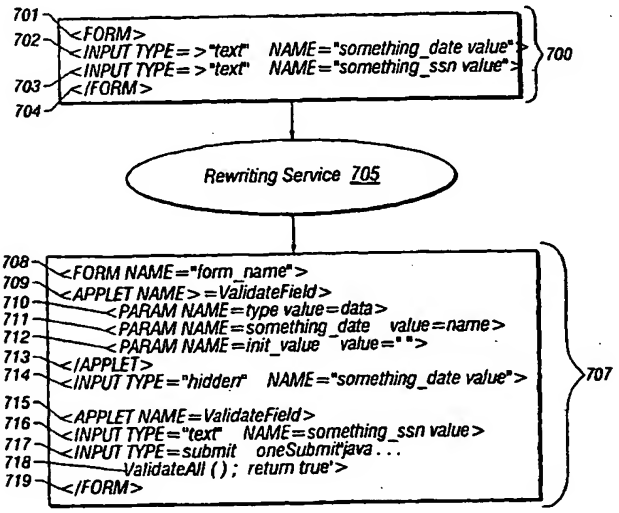
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

[Continued on next page]

(54) Title: METHOD AND APPARATUS FOR COMPLETING A FORM



(57) Abstract: The present invention provides the user with a mechanism for completing a form. When a user issues a request for a form, that request is processed by a rewriting service. In one embodiment of the invention, the rewriting service provides the user with a mechanism for providing data to a form by obtaining data from a data source location. The rewriting service responds to a request for a form by generating a version of the form that contains embedded programs (e.g. applets). The applet elements identify the location of applets configured to obtain data from the data source location. A smart card is an example of a data source location. Access to the smart card may be controlled by the use of an authentication mechanism. The rewritten form is transmitted to a client computer where it utilizes the embedded programs to obtain data from a data source location such as a smart card.

WO 01/025873 A3

(88) Date of publication of the international search report:
11 July 2002

(48) Date of publication of this corrected version:
14 November 2002

(15) Information about Correction:
see PCT Gazette No. 46/2002 of 14 November 2002, Section II

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

METHOD AND APPARATUS COMPLETING A FORM

BACKGROUND OF THE INVENTION

1. FIELD OF THE INVENTION

The present invention relates to computer networks and more particularly to a method and apparatus for completing a form.

2. BACKGROUND ART

Computer users can shop for merchandise and/or information using the Internet. A number of companies maintain web sites where customers can purchase books, records, videos, and many other products. Other web sites provide subscription services that provide the user with access to a service or product, such as channels. Typically, users make purchases or sign up for services by completing forms. A problem with current Internet schemes is that each form must be filled out separately and manually. This is a time consuming and often frustrating process for the user. The product that do allow the user to automatically complete forms do not do so in a secure manner (e.g. IES). For example, mechanisms which do allow a form to be automatically completed if the user visits a form twice (e.g. cookies) are not secure. Such mechanisms also do not allow the user to automatically complete forms from more than one location. Additionally, such mechanisms do not provide users with a way to physically control access to data that is utilized to complete a form.

Current systems do not provide a way for users to quickly complete a form (referred to as "populating" a form) with data. For example, when a user wishes to purchase a product online, the user typically supplies a credit card

number, a shipping address, and any other personal information that may be needed to complete the transaction. The user supplies this information using an input device such as a keyboard to manually enter data into the fields of a form. However, after buying goods from one vendor, for example, the user may wish to purchase goods from a second vendor. The user must manually complete the second vendor's form even though the second vendor may require the purchaser to provide the same kind of information the user provided to the first vendor. This presents a problem to the user because it forces the user to manually enter the same information into different vendor's forms multiple times. Some prior art mechanisms utilize a scheme that provides repeat users with a way to fill out a form with information already provided. However, these schemes do not allow the user to personally control what information is provided to the vendor. Moreover, such techniques do not allow the user to specify how long the data may be kept.

Another problem presented by the prior art is that users do not have a way to securely store sensitive information and then later use that information to populate a form. Current systems do not provide a way to prevent unauthorized users from obtaining access to sensitive information stored on a user's computer. Instead such systems leave data that is entered into a form readily accessible. For example, data that is entered into a form may be stored locally using a technology called cookies. A cookie is a local representation of information related to a particular web page that was previously visited by the user. Cookies are not encrypted by default and may be read by anyone who can obtain them. This is problematic because it unnecessarily exposes sensitive information to unauthorized users. Therefore, there is a need for a method and apparatus for completing a form with data where that data is accessible only to authenticated users.

The problems associated with form population may be better understood by the following discussion of the Internet/World Wide Web, web page creation, embedded forms, cookies, and web browser technology.

The Internet / World Wide Web:

A web browser is a type of computer program that provides users with a mechanism for accessing the World Wide Web (WWW). The WWW is a segment of the Internet comprised of numerous web clients and web servers that communicate with one another using a standard set of protocols. A web server is a computer configured to provide web pages to the web client upon request. A web client typically utilizes the web browser application to request web pages from the web server.

The Internet is a global computer network comprised of an amalgamation of interconnected networks that are capable of readily communicating with one another using a standardized set of protocols. Protocols provide a uniform set of communication parameters that enable computers to effectively transmit and receive data. Most computer networks, including the Internet, utilize several different layers of protocols. Each layer provides the network with different functionality.

The WWW is a segment of the Internet that utilizes an application layer protocol called the HyperText Transfer Protocol (HTTP) to disseminate and to obtain information from users. HTTP is a request/response protocol used with distributed, collaborative, hypermedia information systems. In operation, HTTP enables one computer to communicate with another. For example referring now to Figure 1, web client 100 can use HTTP to communicate with web server 150 via network 125. In this scenario the web server acts as a repository for files 160 and is capable of processing the web client's requests for such files. The files 160

stored on the web server may contain any type of data. For example, the files may contain data used to construct a form, image data, text data, or any other type of data.

HTTP has communication methods that allow web client 100 to request or send data to web server 150. The web client 100 may use web browser 110 to initiate request 115 and receive one or more files from file repository 160. Typically, web browser 110 sends a request 115 for at least one file to web server 150 and the web server forwards the requested file to web client 100 in response 120. The connection is then terminated between web client 100 and web server 150. Web client 100 then uses web browser 110 to display the requested file. A client request 115 therefore, consists of establishing a connection between the web client and the web server using network 125, issuing a response 120 to the request 115, and terminating the connection.

Web server 150 does not maintain any state information about the request once the connection is terminated. HTTP is, therefore, a stateless application protocol. That is, a web client can send several requests to a web server, but each individual request is treated independent of any other request. The web server has no recollection of any previous request. Thus, for example, if a form is completed by the user and submitted to the web server for processing, the web server does not maintain a record of the data entered into the form. One of the key reasons for the success of the HTTP approach is the fact that it is a stateless protocol. However, the stateless nature of the protocol is also what creates the current problem.

Once a file is sent from web server 150 to web client 100 it becomes ready for display. The web client's 100 web browser 110 is typically used to format and display files. Web browser 110 allows the user to request and view a file

without having to learn a complicated command syntax. Examples of several widely used web browsers include Netscape Navigator, Internet Explorer, Opera, and numerous others on PDAs, set top boxes, and screen phones. Some web browsers can display several different types of files. For example, files written using the HyperText Markup Language (HTML), the JavaScript programming language, the ActiveX programming language, or the Portable Document Format (PDF) may be displayed using a web browser. It is also possible to display various other types of files using language such as Standard Generalized Markup Language (SGML) or eXtensible Markup Language (XML).

Creating a Web Page:

A form, which provides one or more places for a user to enter data, can be embedded inside of a web page. A web page may be created using a variety of different data formats and/or programming languages. Most web pages, and as a result most forms, are created using the HyperText Markup Language (HTML). The techniques used to create a web page will now be discussed in further detail.

HTML is a language that may be used to specify the contents of a web page (e.g. web page 220). An HTML description is typically comprised of a set of markup symbols which are described in more detail below. HTML file 250 or any type of data file that contains the markup symbols for web page 220 may be sent to web browser 210. Web browser 210 executing at web client 200 parses the markup symbols in HTML file 250 and produces web page 220, which is then displayed, based on the information in HTML file 250. Web page 220 may contain text, pictures, or forms comprised of embedded text fields, checkboxes, or other types of data that is to be displayed on the web client using web browser 210. Consequently, HTML document 250 defines the web page 220 that

is rendered by web browser 210. For example, the following set of markup symbols directs web browser 210 to display a title, a heading, and an image called "image.jpg":

```
<HTML>
<HEAD>
<TITLE> This is a document title </TITLE>
</HEAD>
<BODY>
<H1> This text uses heading level one </H1>
<IMG SRC="http://www.sun.com/image.jpg">
</BODY>
</HTML>
```

In the above example, markup symbols (e.g. "<" and ">") indicate where each HTML command (e.g. TITLE) begins and ends. An HTML command, which is typically surround by markup symbols, provides the web browser with instructions to execute. Markup symbols typically surround an HTML command. The "<" symbol indicates the start of an HTML command an the "</" symbol indicates the end of an HTML command. Each start or end command has a corresponding ">" to indicate the close of that particular command. Information associated with the HTML command may be contained within the HTML command's start and end symbols. An HTML command is used to by the web browser 210 to determine how to process the block of information associated with the two commands.

In the above example, "<TITLE>", and "</TITLE>" are examples of HTML commands surrounded by markup symbols. The "</TITLE>" HTML command directs web browser 210 to place the text "This is a document title" in the title bar of web browser 210.

Some HTML commands have attribute names associated with the command. For example, HTML command "<IMG>", directs web browser 210 to display an image. A "SRC=" attribute identifies the location and name of the image to be displayed. In the above example, the statement "<IMG

SRC="http://www.sun.com/ image.jpg">" tells the web browser to display an image named "image.jpg" that can be obtained from the web server located at "http://www.sun.com."

Embedding a Form into a Web Page

An HTML file may also contain HTML commands that cause the web browser to render a web page that contains fields for entering data. The portion of the web page that contains the data entry fields may be referred to as a form. When the web page is comprised of primarily data entry fields the entire web page is sometimes also referred to as a form. As is discussed below, HTML includes an HTML form command that may cause the browser to display data entry fields. A text box, a drop down menu, a check box, a command button, a toggle button, or any other kind of interface component capable of receiving input are some examples of the type of data entry fields that may be placed in a form. Data is manually entered into the fields by the user and then submitted to a web server for processing. As previously discussed, when a user wishes to use a form to purchase a product, for example, the user manually fills in text boxes located on the form with the type of information needed to carry out the transaction. Other types of data entry fields require the user to select from one or more options such as in the case of a menu, a toggle button, or a command button, for example. When the form is complete the user submits the completed form to the vendor's web server by pressing a command button, for example.

A problem with collecting information from the user in this manner is that the user is required to manually enter data into the fields of a form. This takes time and becomes unnecessarily burdensome when the user wishes to complete more than one form with the same information.

Figure 3 provides an example of, a form created using the HTML definition language. Code block 310 contains HTML command examples. When a document comprising code block 310 is transmitted to web browser 300 executing on web client 330, it causes form 305 to be displayed. Web browser 300 displays form 305 by parsing the HTML commands contained in code block 310 and then using the information obtained to format form 305. Once the user finishes entering information into form 305, the user may submit the form by pressing command button 331. Pressing command button 331 sends the information entered by the user in form 305 to web server 320 using either the GET or the POST method.

The <FORM> command shown in code block 310 indicates the beginning of a form. Once the initial FORM command is placed into the HTML document other HTML commands may be entered between the initial FORM command and the closing FORM command (e.g. </FORM>) that represent, for example, one or more data entry fields such as a text-box, drop-down lists, check boxes, radio buttons, and input buttons. The INPUT command is used to specify different types of data entry fields. The type of data entry field created is dependant upon the value assigned to the INPUT command's TYPE attribute. For example, the INPUT command can create a text-box, a password field, a hidden field, a button, a radio button, or a checkbox. A "text" value assigned to the TYPE attribute of an INPUT command creates a text box, for example. Similarly, a TYPE of "checkbox" creates a checkbox. The following code, if inserted after the FORM element, creates a text box and a checkbox:

```
<INPUT TYPE="text" NAME="user_name">
<INPUT TYPE="checkbox" NAME="user_item1">
```

The HTML tags and text contained in code block 310, for example, create form 305 when displayed using web browser 300.

A <FORM> tag may contain an ACTION and/or METHOD attribute. The ACTION attribute specifies what program to execute when form 305 is submitted for processing. The METHOD attribute describes how data entered into the form is handled. There are two METHOD choices: GET and POST. Each one of them is capable of processing the data entered into a form. To illustrate, if a user named Duke who resides at 123 Main Street, Ca 90211 completes the Name and Address fields shown on form 305 by filling in spaces 345-347, then the following data string 350 is created and sent to web server 320.

user_name=Duke&user_address=123_Main Street,_CA_90211

This data string is handled by either the GET method or the POST method. The GET method appends information entered into form 305 by the user onto the end of a Uniform Resource Locator (URL 335) or in a QUERY_STRING environment variable. URL 335 is submitted to web server 320 by web client 330 in an HTTP request 315. Web server 320 is responsible for processing the submitted information. A problem with the GET method is that the information contained in URL 335 is not stored on web client 330 for later use and cannot therefore be later used to populate form 305. With the POST method, the information placed into form 305 by the user is placed into the data stream of the HTTP protocol. This allows web server 320 to process form 305 data using the standard input data stream. Standard input is a method for providing a computer with data that is well know to those skilled in the art. The POST method also does not provide an effective way to easily maintain and store data on web client 330 for later use without doing extra work using the Common Gateway Interface (CGI).

The ACTION attribute of the FORM tag indicates where the computer program tasked with processing data string 350 resides. For example, the following portion of code block 310 specifies that a computer program named form_processor.pl is located in cgi-bin 321 of sever 320.

```
<FORM METHOD=POST ACTION=http://server320
/form_processor.pl>
```

The form_processor.pl program is responsible for processing data string 350. Programs that utilize CGI standard are typically located in a common directory (e.g. cgi-bin 321). CGI is a specified standard for communicating between HTTP servers and server-side gateway programs. A CGI program is capable of storing data supplied by the user, but it cannot populate a form for a user located at a client computer (e.g. web client 330). CGI programs execute at the server and are therefore incapable of providing data to a form at the client.

The program "form_processor.pl" used in the above example is an example of a server-side gateway program that processes data input by the user via form 305, for example. When form 305 is submitted web server 320 executes the "form_processor.pl" program and passes the data that was entered by the user and then sent from web client 330 to server 321. That is, data string 350 is passed to "form_processor.pl" when a form is submitted. When the gateway program is done processing data string 350 the result is sent to server 320 and a response may be forwarded back to web client 330. Gateway programs can be compiled programs written in languages such as C or C++ or they can be executable scripts written using a scripting language such as PERL, TCL, or various other language. Once data string 350 is processed it may be stored on server 320 in data store 322.

Thus, existing mechanisms for processing forms reside on the server and are unable to populate a form on the client (e.g. data resides on the server and programs that process data execute on the server).

HTTP Cookies

One example of data that may be stored on the client is a cookie. A cookie is an HTTP header that consists of a text-only string. The text-string is entered into the memory of a web client and accessible by the web browser. This text-string is initially set by the computer serving the web page and consists of the domain name, path, lifetime, and value of a variable that the server sets. If the lifetime of this variable is longer than the time the user spends at that site, then the text-string is saved on the web client for later use. As a result, cookies provide a way to store and later recall values entered into a web page by the user. For example, cookies allow a web server to individually customize a web page for each user who visits the page.

A cookie may be used to populate the elements of a form that a user has previously completed. For example, referring now to Figure 9, if a user completes field 945-947 of form 905 and submits it to server 920 via submission path 915, server 920 can place the data submitted by the user in cookie 931 by sending web client 910 data via path 925. If the user returns to form 905 again Web browser 930 executing at web client 910 may use the data saved in cookie 931 populate fields 945-947 with the data previously entered by the user. Thus, a cookie can free the user from having to complete form 905 more than once. Cookies can be used to fill forms the user has previously visited. However, the data that is stored by the cookie cannot be used to populate other forms. That is, a problem with cookies is that forms located on different servers are not

provided access to the same cookie. Only web servers that are listed as having permission to access a particular cookie may obtain such access.

A cookie is only accessible to the server that set, or created it. Thus, a second form cannot use the first form's cookies to obtain the user's name and address. For example, the web server located at www.merchantA.com may be allowed to read a cookie, but the web server located at www.merchantB.com may not be allowed to read the cookie created by www.merchantA.com. This loads the disk full of redundant information. Moreover, users are not sure what type of information is stored in such cookies.

Referring now to Figure 4, the process used to create and retrieve a cookie is illustrated. Initially, web browser 401 which resides on web client 400 issues a request 410 for web page 415 from web server 450. Web server 450 responds by sending a copy of web page 415 to web client 400 via response 411. Web client 400 uses web browser 401 to display web page 415 to the user. To create a cookie 460, web server 450 sends a "Set-Cookie" command in the header line contained in response 411. For example, in response to a request 410, web server 450 may send the following command in HTTP response 411:

Set-Cookie: NAME=VALUE; expires=DATE; path=PATH; domain=DOMAIN_NAME; secure.

The NAME and VALUE parameter contain the information included in the cookie. DATE is the time at which the cookie expires and is thus no longer saved on web client 210. DOMAIN is the host address or domain name for which the cookie is valid. For example, if a cookie is set by a computer having a domain name of www.merchantA.com, only the server responsible for that particular domain name is provided access to the cookie. The PATH parameter specifies a subset of URL's at the appropriate domain for which the cookie is valid. If the keyword secure is used then the cookie is only transmitted over a secure

connection. All of these parameters except the NAME=VALUE are optional to set a cookie. Once the Set-Cookie command is sent to web client 400 a cookie 460 is placed on web client 400.

To create a cookie 460 using the UNIX operating system, for example, web server 450 could execute the following shell script in response to a request 410:

```
#!/bin/sh
echo Content-type: text/html
echo Set-cookie: FooBar=foo; expires=Wednesday,
        02-11-99 12:00:00 GMT
echo
echo <H1>A Cookie named Foobar containing the text foo is now present
on your computer</H>
```

This script stores "FooBar=foo" on web client 400. The following script allows web server 450 to read the HTTP cookie:

```
#!/bin/sh
echo Content-type: text/html
echo
echo The data supplied here was obtained from a
cookie:<P>
echo $HTTP_COOKIE
```

Once cookie 460 is created, the information stored in the cookie can be used in many different ways. Information in cookie 460 may be accessed by a script that has authorization to insert cookie 460 data into a form, for example. However, a problem with cookies is that no authentication mechanism is required to obtain the information they store. A cookie is a text file and is not stored in encrypted form. While a browser may limit access to a cookie, the text that is stored in the cookie may be accessed or read by any program that can read a text file. Therefore, it is not wise to store any sensitive data in a cookie.

A further problem is that unauthorized users may review another user's cookie files and thereby ascertain information about that user's spending, purchasing, and/or web browsing habits. Once such information is obtained it may then be utilized to target the user for theft and/or marketing.

Predefined Paste Operation

Some web browsers provide users with a mechanism for pasting predefined data into a form. Under the generic preferences menu of the Opera web browser, for example, the user can select a command button labeled personal information. When this command button is selected a dialog box that contains fields for entering personal information opens. If the user desires, the user can enter a name, an address, a phone number, an e-mail address, and/or any other kind of information into the text fields of the dialog box.

Once the user enters information into the fields of the dialog box and clicks "OK" the information that was entered can be pasted into the fields of a form by performing a right click operation. For example, if the user encounters a form that has a place for entering an address, the user can elect to paste the address previously specified in the dialog box by first selecting the address field and then right-clicking on a pointing device such as a mouse and selecting from a drop down menu the item titled "insert address".

A problem with pasting information into the fields of a form in this manner is that the user is required to manually select the place to insert the information and the type of information to insert into that place. Another problem is that unauthorized users are not prevented from accessing the information entered into the dialog box.

SUMMARY OF THE INVENTION

The present invention provides the user with a mechanism for completing a form. When a user issues a request for a form, that request is processed by a rewriting service. In one embodiment of the invention, the rewriting service provides the user with a mechanism for providing data to a form by obtaining data from a data source location. The rewriting service responds to a request for a form by generating a rewritten version of the form that contains embedded programs (e.g. applets). The applet elements identify the location of applets configured to obtain data from the data source location. A smart card is an example of a data source location. Access to the smart card may be controlled by the use of an authentication mechanism. The rewritten form is transmitted to a client computer where it utilizes the embedded programs to obtain data from a data source location such as a smart card.

The rewriting service obtains a requested form and substitutes tags of one type for tags of another type. For example, the rewriting service may substitute tags that satisfy a set of predetermined criteria for tags that identify the location of an executable program. In one embodiment of the invention, this is accomplished by replacing XML elements (e.g. tags) that are inserted into the requested form in a standardized format with APPLET elements that identify the location of an executable applet program. This enables browsers that are not capable of parsing XML to utilize the advantages of the present invention. The rewriting service can replace elements of one type with elements of a second type. For example, any of the elements contained in a form (e.g. an HTML document) may be replaced with APPLET elements. FORM tags or INPUT tags, for example, may also be replaced with APPLET elements.

The APPLET elements identify the location of an applet program configured to obtain data from a data source. A smart card or other type of data store is an example of a data source.

One embodiment of the invention contemplates the user of an authentication mechanism to provide the user with control over the type of information utilized to complete a form. When the applet initially seeks to obtain data from a data source it may prompt the user to insert a smart card. When the user inserts the smart card the user may be prompted for authentication information. If the user enters the correct information, the applet is allowed to read data from the smart card. If the user does not enter the correct information then access to the data stored on the smart card is not allowed.

One embodiment of the invention comprises computer code configured to advertise services offered by the proxy, the rewriting service, the applets, the server computer, the card server, or the client computer. A service is an entity that can be used by a user, a program, or another service to perform a certain kind of predetermined functionality. The ability to dynamically advertise services provides a mechanism to personalize the tasks performed by a service. The user, for example, could customize the way a particular type of form is to be completed. A user that wants to be prompted before the system disclose his social security number could elect to have this event occur whenever a form is encountered that prompts the user for a social security number.

DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of the various components used by the World Wide Web.

Figure 2 is a block diagram illustrating the components used to create a web page.

Figure 3 is a block diagram illustrating how web client interacts with a web server to display a form.

Figure 4 illustrates the process used to create and retrieve a cookie.

Figure 5 is an illustration of the components used by one embodiment of the invention to provide the user with a mechanism for completing a form.

Figure 6 is a flow chart illustrating the steps performed by one embodiment of invention.

Figure 7 is a block diagram illustrating the input transmitted to the rewriting service and the output generated by the rewriting service in one embodiment of the invention.

Figure 8 is an example of a form used by one embodiment of the invention to collect data.

DETAILED DESCRIPTION

A method and apparatus for completing a form is described. In the following description numerous specific details are set forth in order to provide a more thorough understanding of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without these specific details. In other instances, well-known features have not been described in detail so as not to obscure the invention.

The present invention provides the user with a mechanism for completing a form. In one embodiment of the invention, a rewriting service, executing at the server, responds to a request for a form by generating a version of the form that contains embedded programs (e.g. applets). The rewritten form is transmitted to a client computer where it obtains data from a data source location such as a smart card.

The rewriting service provides the user with a mechanism for providing data to a form by obtaining data from a data source location. In one embodiment of the invention, applet elements are inserted by the rewriting service. The applet elements identify the location of applets configured to obtain data from the data source location. A smart card is an example of a data source location. Access to the smart card may be controlled by the use of an authentication mechanism.

Generating a Form:

A form is a type of digital document that contains one or more data receptacles. It is possible to create a form using a variety of different computer programming languages. The invention contemplates the use of forms created

using HyperText Markup Language (HTML), eXtensible Markup Language (XML), Standardized General Markup Language (SGML), JavaScript, the Java programming language, or any other programming language capable of generating a document that provides a location for the user to enter data (e.g. a data receptacle). For example, a spreadsheet, a word processing document, , a database table, or any other type of document having one or more data receptacles may be referred to as a form. It is also possible to create a form using high-level programming languages such as C/C++, Pascal, or Smalltalk. Forms are typically generated by a computer program and presented to the user via a Graphical User Interface (GUI). For example, a web browser, an operating system, or any other type of computer program may be utilized to display a form to the user. In one embodiment of the invention, any type of computer program capable of executing a layout engine can generate a form. For example, the NGLayout engine which is based on open Internet standards such as HTML 4.0, CSS 1/2, XML 1.0, and the Document Object Model can be embedded into various execution environments and used to generate a form. A form may be pre-created and stored on a server computer until a request for the form is issued. However, it is also feasible to store the form locally. For example, a form may reside on a client computer instead of the server computer. In either case, the form is delivered to the user when a request is issued. In accordance with one embodiment of the invention, a form is not generated until it is requested by the user. This is referred to as generating a form dynamically or "on the fly" To generate a form in such a manner the server contains computer program code configured to generate a form. When the user initiates a request for a form, the server executes the computer program code and thereby generates a form. The form is then transmitted to the client for display.

Data Receptacles:

In one embodiment of the invention, each form contains one or more data receptacles. Data receptacles are designed to obtain and/or receive information. Information may be placed inside of a data receptacle by the user or by a computer program configured to perform an insert operation. For example, it is possible to populate a data receptacle with text or numerical information. Data receptacles may also be referred to as fields, form elements, or text boxes.

Referring now to Figure 8, an example of a form used to collect data is shown. Form 800 may be displayed to the user in a web browser 801. Form 800 utilizes data receptacles 804-812 to collect information about a user. In one embodiment of the invention, form 800 is embedded inside of a web browser 801. Data receptacles 804-807 may be used to collect billing information such as the name, company, address, phone number, and e-mail address of the party responsible for payment. Data receptacles 808-811 may be used to collect shipping information when the shipping information is different than the billing information. Data receptacle 812 is intended to collect credit card or other payment information. One embodiment of the invention provides a way to populate data receptacles with data without requiring the user to manually enter the data.

Overview of the Form Completion Process:

Figure 5 is an illustration of the components used by one embodiment of the invention to provide the user with a mechanism for completing a form. A request 503 for form 505 is typically initiated at client computer 500 by a computer program capable of displaying form 505 to the user. A web browser 507, for example, may transmit request 503 to server 509 via interconnection fabric 511. When server 509 receives request 503, it may be handed to proxy 515

for processing. In one embodiment of the invention, proxy 515 executes rewriting service 517. However, request 503 may also be handed directly to rewriting service 517 and the rewriting service may execute without proxy 515. Rewriting service 517 executes a resynchronization process that obtains form 505 from data store 590 and modifies it. Thus, form 505 becomes rewritten as form 506.

The modified version of form 505 (e.g. form 506) contains embedded code configured to interact with card server 519. The rewriting service, for example, may replace existing HTML FORM elements with APPLET elements and thereby provide a way for client computer 500 to obtain applets 525-528 from server computer 509. When rewriting service 517 generates form 506, server 509 may transmit form 506 to client 500 via response 556. Response 556 may be sent across interconnection fabric 511 to client 500. Once the rewritten form resides on client 500, the applets identified in the APPLET elements may be executed. In one embodiment of the invention, applets 525-528 are obtained from data store 590 located on server 509. However, applets 525-528 may be obtained from any source connected to interconnection fabric 511.

Applets 525-528 execute in web browser 507 and communicate with card server 519 to obtain data 521 from a data source location. For example, card server 519 may be configured to read data 521 from smart card 523. Data 521, however, may also reside on other locations, such as server 509. In one embodiment of the invention, data 521 is obtained from smart card 523 and sent to web browser 507 for display in data receptacles 529-532. Data receptacles 529-532 and the information placed in them become visible to the user when form 506 is shown on display 550.

Figure 6 is a flow chart illustrating one embodiment of invention. At step 600, a request for a form is issued. A request is typically initiated by the user of a client computer, but may also be initiated by computer code, executing at the client, configured to automatically request a form when an event occurs. For instance, a form prompting the user for registration info (e.g. name, address, etc...) may be automatically presented to the user when the user installs a new computer program. The invention also contemplates the use of a web browser to request a form. The entity that initially issues the request is referred to as a requestor.

In one embodiment of the invention, the requested form contains XML tags embedded into it. However, the form may also contain data written in other programming languages such as HTML, SGML, or any other language compatible with the protocols utilized to transmit data across a computer network. The requested form resides on a source computer. The source computer may be a server computer, a client computer, or any other computer accessible via a computer network. In one embodiment of the invention, the requested form is processed by a rewriting service before it is forwarded to the requestor (e.g. the web browser). The rewriting service typically resides on the server computer, but may be placed on any computer capable of functioning as a source computer. Once a form is requested, step 603 executes and the source computer obtains the requested form from the source computer.

At step 605, the requested form is forwarded to a rewriting service. In one embodiment of the invention, the requested form is transmitted across a computer network to the computer containing the source computer. However, If the rewriting service resides on the source computer, the requested form can be forwarded directly to the rewriting service without having to be transmitted across a computer network. Once the rewriting service receives the requested

form it executes step 607 where it processes the form and thereby prepares the form for transmission to the requestor. An additional possibility is that the supplied form can be signed using the rewriting service. At step 609, the rewritten form is transmitted to the requestor for display. For example, the rewritten form may be sent to a web browser which parses the information comprising the form to generate an image for display to the user. If, for example, the rewriting service resides on the server computer, the rewritten form is transmitted across a computer network to the client computer for display. If, the rewriting service does not reside on the server computer, but instead resides on the client, then the rewritten form is sent directly to the program responsible for displaying the form to the user (e.g. the web browser).

Displaying the rewritten form causes step 611 to execute. At step 611, the APPLET elements encompassed into the rewritten form are parsed. In one embodiment of the invention, this causes the client computer to obtain and execute the applets referred to in the APPLET element. For example, if one of the APPLET elements contained in the rewritten form refer to an applet named ValidateField, the ValidateField applet will execute. The invention contemplates the use of applets configured to obtain data from a data source (e.g. at step 613). For example, in one embodiment of the invention, the applets acquire data from a smart card server configured to communicate with a smart card that is inserted into the client computer by the user. However, the applets may be configured to obtain data from other types of data sources made available to the client computer by a computer network. In one embodiment of the invention, access to the data source is not permitted until the user is authenticated. For example, the user may be required to enter a user name and password before data is retrieved from the smart card. Other authentication mechanisms, such as encryption may also be integrated into the authentication process. At step 615,

the invention places the data accumulated from the data source location into one or more of the data receptacles present on the form initially requested by the requestor and it posts the form back to the server.

The Rewriting Service:

When a user issues a request for a form, that request is processed by a rewriting service. The rewriting service obtains a requested form and substitutes tags of one type for tags of another type. For example, the rewriting service may substitute tags that satisfy a set of predetermined criteria for tags that identify the location of an executable program. In one embodiment of the invention, this is accomplished by replacing XML elements (e.g. tags) that are inserted into the requested form in a standardized format with APPLET elements that identify the location of an executable applet program. The rewriting service can replace elements of one type with elements of a second type. For example, any of the elements contained in a form (e.g. an HTML document) may be replaced with APPLET elements. FORM tags or INPUT tags, for example, may also be replaced with APPLET elements.

For example, referring now to Figure 7, if code block 700 is processed by rewriting service 705, code block 707 is generated. Line 701, for example, becomes line 708. Thus, <FORM> is translated into <form name = form_name> once it is processed by the rewriting service. The INPUT tags, shown at lines 702 and 703 become parameters for an applet. The NAME parameter associated with the INPUT tag, for example, is inserted as a parameter for an applet named ValidateField. As a result, line 702 which reads <INPUT TYPE="text" NAME="something_date value"> becomes lines 709-713 which reads

```
<APPLET NAME="ValidateField">
    <PARAM NAME=type VALUE=data>
    <PARAM NAME=something_date value=name>
    <PARAM NAME=init_value value="">
</APPLET>.
```

The NAME parameter in line 702 may also be inserted into code block 707 as a hidden field. Line 714 is an example of a hidden field. When an INPUT value is passed a type parameter of hidden, nothing is displayed to the user. However, the hidden value is accessible by a computer program configured to process the form. Not all lines processed by the rewriting service are rewritten. Line 703, for example, remains unaltered by the rewriting service. Thus, line 703 is identical to line 716. The rewriting service may also partially alter lines in the originally requested form. In one embodiment of the invention, the rewriting service is configured to insert additional lines. Therefore, the rewritten form may contain more or less lines than originally in the requested form. Once the rewriting service completes the rewriting process, the form that is generated is transmitted to the requestor (e.g. the client computer) for display. At this point, the applets referred to in the APPLET elements may execute.

Extensible Markup Language (XML) is a technology that provides a way to separate the value of the data from the presentation of the data. The present invention also works with XML. Using XML the HTML portion of the form can be populated with XML tags.

The Applet:

An applet is a small computer program configured to perform one or more predefined tasks. In one embodiment of the invention, applets obtain data from a data source location and insert that data into one or more data receptacles on behalf of a user. For example, once the rewriting service has properly embedded information into the web page, the applet can populate a

data receptacle with the information needed to complete a commercial transaction. An applet may obtain data from a variety of different sources. Any type of static, dynamic, random access, or read only memory device may be programmed to retain the type of data utilized by the rewriting service.

A smart card or flash memory device, for example, can store such data. Other memory devices such as a hard drive, a CD-ROM, a CD-R, a DAT, or a DVD device can also retain data for use by the applet. The applet may access the memory device locally or over a communication medium such as a computer network. In one embodiment of the invention, the memory devices contain personal information about a user. However, it is feasible for the applet to process any kind of data. The applet, for example, may also process business data and/or credit card data.

Authentication:

The applets may be configured to obtain data from a smart card (e.g. the data source location). When the user inserts a smart card the smart card communicates with a card server. In one embodiment of the invention, the card server resides on the user's system. However, the card server may be placed on any system accessible via a computer network. Access card is not permitted unless the user inserting the smart card is authenticated. The present invention contemplates the use of various authentication mechanisms. For example, the user may be required to enter a user name and password before the card server is permitted to read data from the smart card. Various other techniques for ensuring the data on the smart card is only provided authorized sources may also be employed. A PIN number or similar method of authenticating the user, for example, may also be utilized to ensure that only authorized access to the

smart card occurs. Biometric instruments may also be utilized to determine if a person is authorized to access the smart card.

The authentication mechanism provides the user with control over the type of information utilized to complete a form. When the applet initially seeks to obtain data from a data source it may prompt the user to insert a smart card. When the user inserts the smart card the user may be prompted for authentication information. If the user enters the correct information, the applet is allowed to read data from the smart card. If the user does not enter the correct information then access to the data stored on the smart card is not allowed.

Advertising Services:

One embodiment of the invention comprises computer code configured to advertise the services offered by the proxy, the rewriting service, the applets, the server computer, the card server, or the client computer. A service is an entity that can be used by a user, a program, or another service to perform a certain kind of predetermined functionality. The Jini ™ architecture provides a mechanism for integrating services into a single, dynamic distributed system. Services, for example, are capable of providing computational functionality, storage, a communication channel to another user, a software filter, access to a hardware device, or another user.

Members of a Jini ™ system federate in order to share access to services. For example, if the card server joins the federation it could provide the functionality it offers to another service. Members of a Jini ™ system are capable of utilizing the services each member offers to collectively perform a particular task. For example, services may make use of another service, and a client of one service may itself be a service for clients of its own.

The rewriting service, for example, may be configured to interact with the card server to perform a task for the user. In one embodiment of the invention, the ability to dynamically advertise services provides a mechanism to personalize the tasks performed by a service. The user, for example, could customize the way a particular type of form is to be completed. A user that wants to be prompted before the system discloses his social security number could elect to have this event occur whenever a form is encountered that prompts the user for a social security number.

Thus, a method and apparatus for completing a form is described.

CLAIMS

1. A method for completing a form comprising:
initiating a request for a form having at least one data receptacle;
obtaining said form in response to said request;
providing said form to a rewriting service.

2. The method of claim 1 wherein said rewriting service generates a rewritten form by substituting a first type element in said form for a second type element.

3. The method of claim 2 wherein said second type element identifies the location of program code.

4. The method of claim 2 further comprising:
parsing said rewritten form;
displaying said rewritten form to a user.

5. The method of claim 4 further comprising:
determining whether said user is authenticated;
obtaining data from a data source location when said user is authenticated;
providing said data to said at least one data receptacle.

6. The method of claim 5 wherein said data source location is smart card.

7. The method of claim 6 wherein a card server interacts with said smart card.

8. The method of claim 1 wherein said rewriting service executes on a server.

9. The method of claim 3 wherein said program code is an applet.

10. The method of claim 5 wherein said rewriting service offers services that allow said user to customize when said step of providing data to said at least one data receptacle occurs.

11. The method of claim 7 wherein said card server offers services to said rewriting service.

12. A computer program product comprising:

a computer usable medium having computer readable program code embodied therein for completing a form, said computer program product comprising:

computer readable program code configured to obtain a form having data receptacles in response to a request;

computer readable program code configured to generate a rewritten form by substituting a first type element in said form for a second type element.

13. The computer program product of claim 12 wherein said second type element identifies the location of program code.

14. The computer program product of claim 12 further comprising:

computer readable program code configured to provide said rewritten form to a requestor.

computer readable program code configured to parse said rewritten form at said requestor;

computer readable program code configured to display said rewritten form to a user at said requestor.

15. The computer program product of claim 14 further comprising:

computer readable program code configured to determine whether said user is authenticated;

computer readable program code configured to obtain data from a data source location when said user is authenticated;

computer readable program code configured to provide said data to said at least one data receptacle.

16. The computer program product of claim 15 wherein said data source location is smart card.

17. The computer program product of claim 16 wherein a card server interacts with said smart card.

18. The computer program product of claim 13 wherein said program code is an applet.

19. The computer program product of claim 15 further comprising:
computer readable program code configured to offer services that allow said user to customize when said computer readable program code provides said data to said at least one data receptacle.

20. A method for completing a form comprising:
initiating a request for a form having at least one data receptacle;
obtaining said form in response to said request;
providing said form to a rewriting service;
executing said rewriting service to generate a rewritten form by substituting a first type element in said form for a second type element;

21. The method of claim 20 wherein said second type element identifies the location of program code.

22. The method of claim 20 further comprising:
parsing said rewritten form;
displaying said rewritten form to a user;

23. The method of claim 22 further comprising:
determining if said user is authenticated;
obtaining data from a data source location when said user is authenticated;
providing said data to said at least one data receptacle.

24. The method of claim 23 wherein said data source location is smart card.

25. The method of claim 24 wherein a card server interacts with said smart card.

26. The method of claim 20 wherein said rewriting service executes on a server.

27. The method of claim 21 wherein said program code is an applet.

28. The method of claim 23 wherein said rewriting service offers services that allow said user to customize when said step of providing data to said at least one data receptacle occurs.

29. A method for completing a form comprising:

initiating a request for a form having at least one data receptacle;

obtaining a rewritten form having program code, said program code configured to communicate with a data source location having data;

parsing said rewritten form;

executing said program code to obtain said data from said data source location;

displaying said rewritten form containing said data to a user;

30. The method of claim 29 wherein said rewriting service generates said rewritten form by substituting a first type element in said form for a second type element.

31. The method of claim 30 wherein said second type element identifies the location of said program code.

32. A method for completing a form comprising:

responding to a request for a form by obtaining a form;

providing said form to a rewriting service, said rewriting service generating a rewritten form by substituting a first type element in said form for a second type element;

transmitting said rewritten form to a client computer for processing.

33. The method of claim 32 wherein said second type element identifies the location of said program code, said program code configured to obtain data from a data source connected to said client computer.

34. A method for completing a form comprising:

a means for initiating a request for a form having at least one data receptacle;

a means for obtaining said form in response to said request;

a means for providing said form to a rewriting service;

a means for executing said rewriting service to generate a rewritten form by substituting a first type element in said form for a second type element.

35. The method of claim 34 wherein said second type element identifies the location of program code.

36. The method of claim 34 further comprising:

a means for parsing said rewritten form;

a means for displaying said rewritten form to a user.

37. The method of claim 36 further comprising:

a means for determining if said user is authenticated;

a means for obtaining data from a data source location when said user is authenticated;

a means for providing said data to said at least one data receptacle.

38. The method of claim 34 wherein said data source location is smart card.

39. The method of claim 34 wherein a card server interacts with said smart card.

40. The method of claim 34 wherein said rewriting service executes on a server.

41. The method of claim 35 wherein said program code is an applet.

42. The method of claim 37 wherein said rewriting service offers services that allow said user to customize when said step of providing data to said at least one data receptacle occurs.

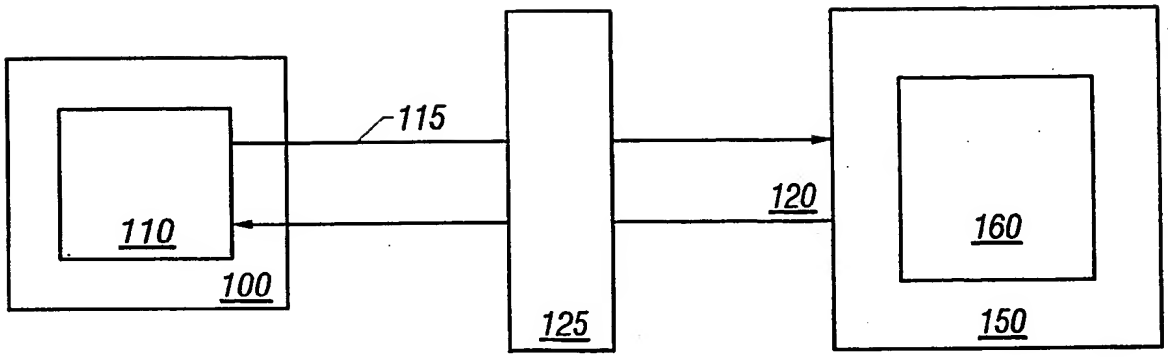


Figure 1

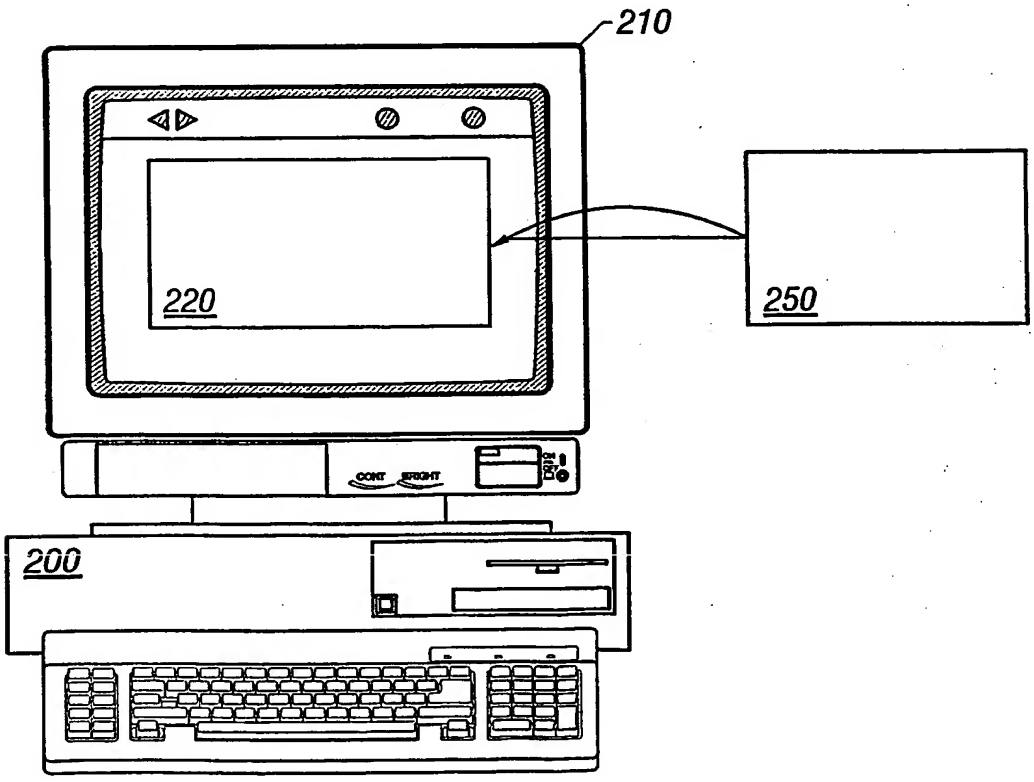


Figure 2

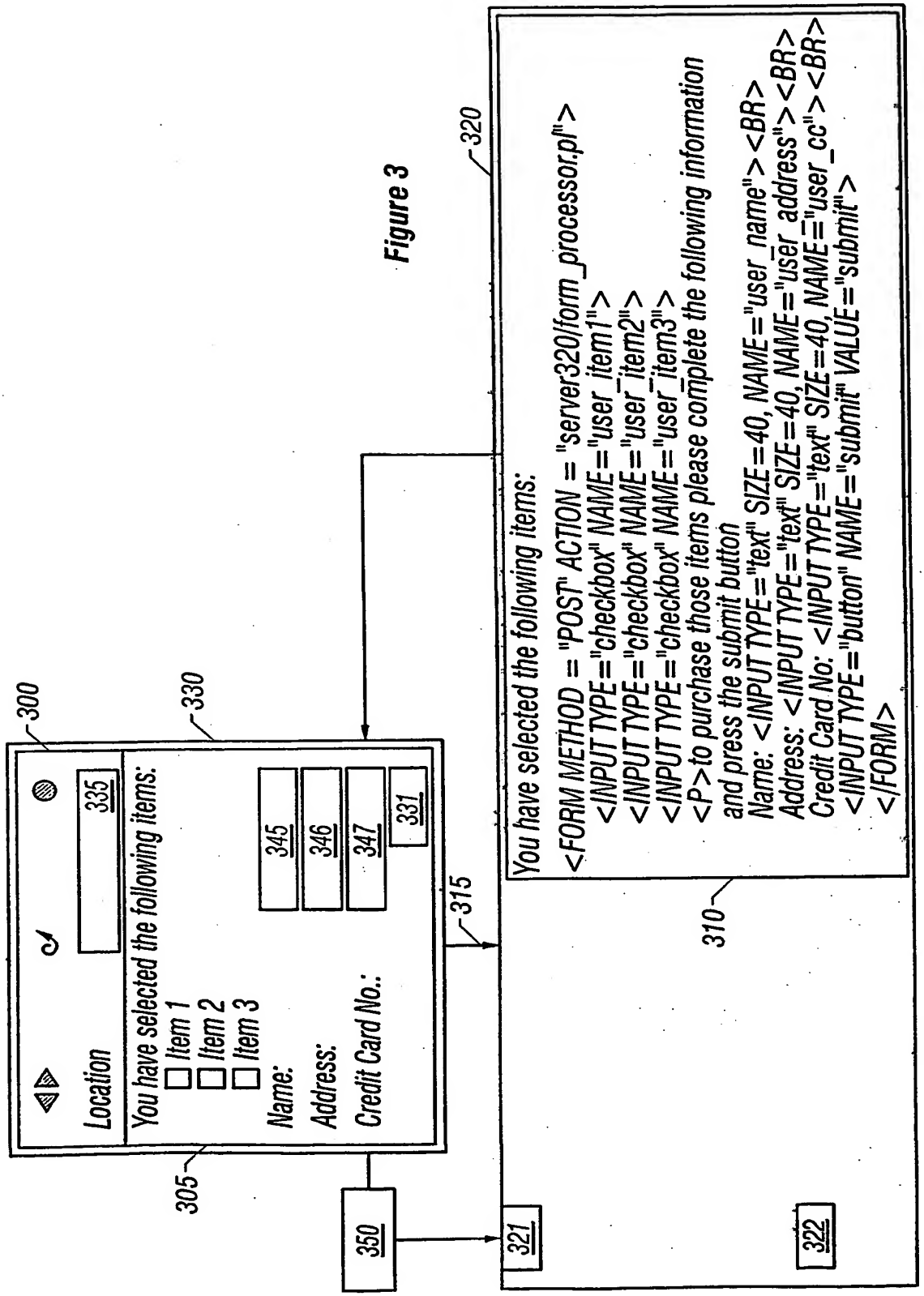


Figure 3

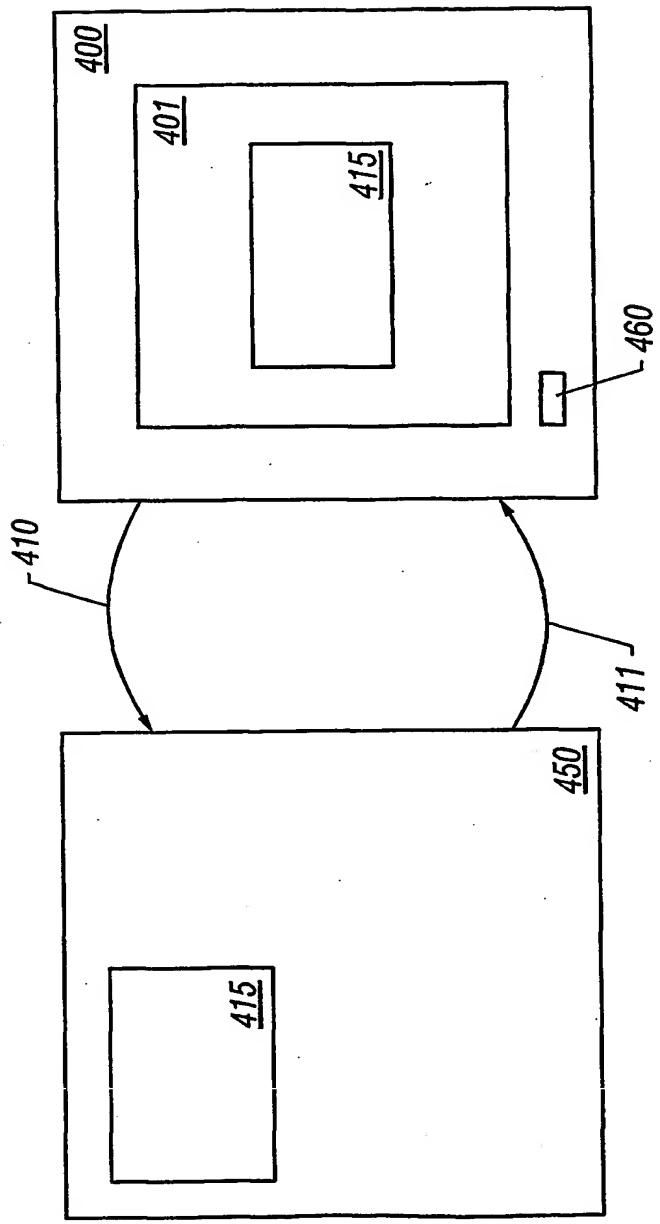


Figure 4

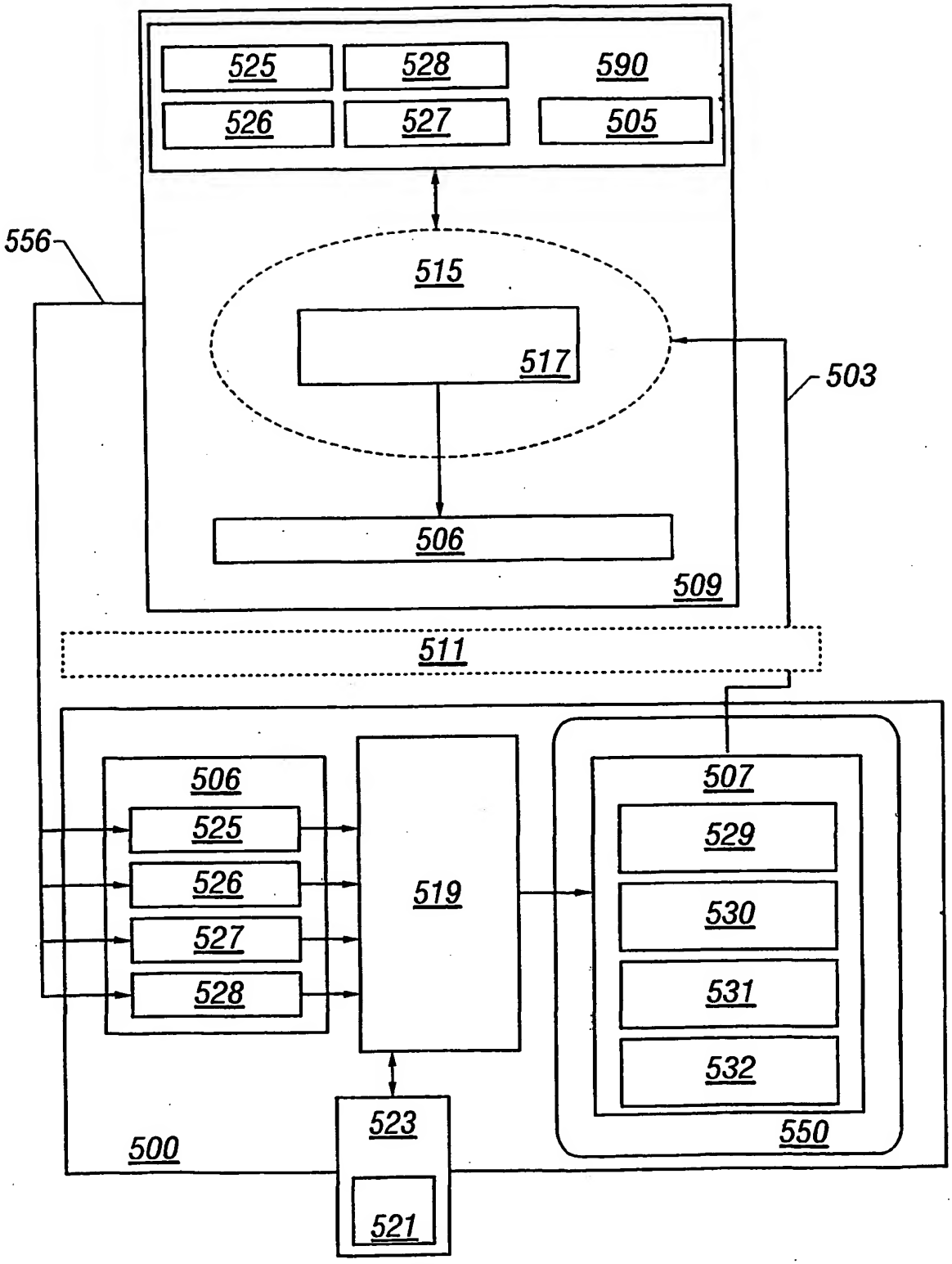


Figure 5

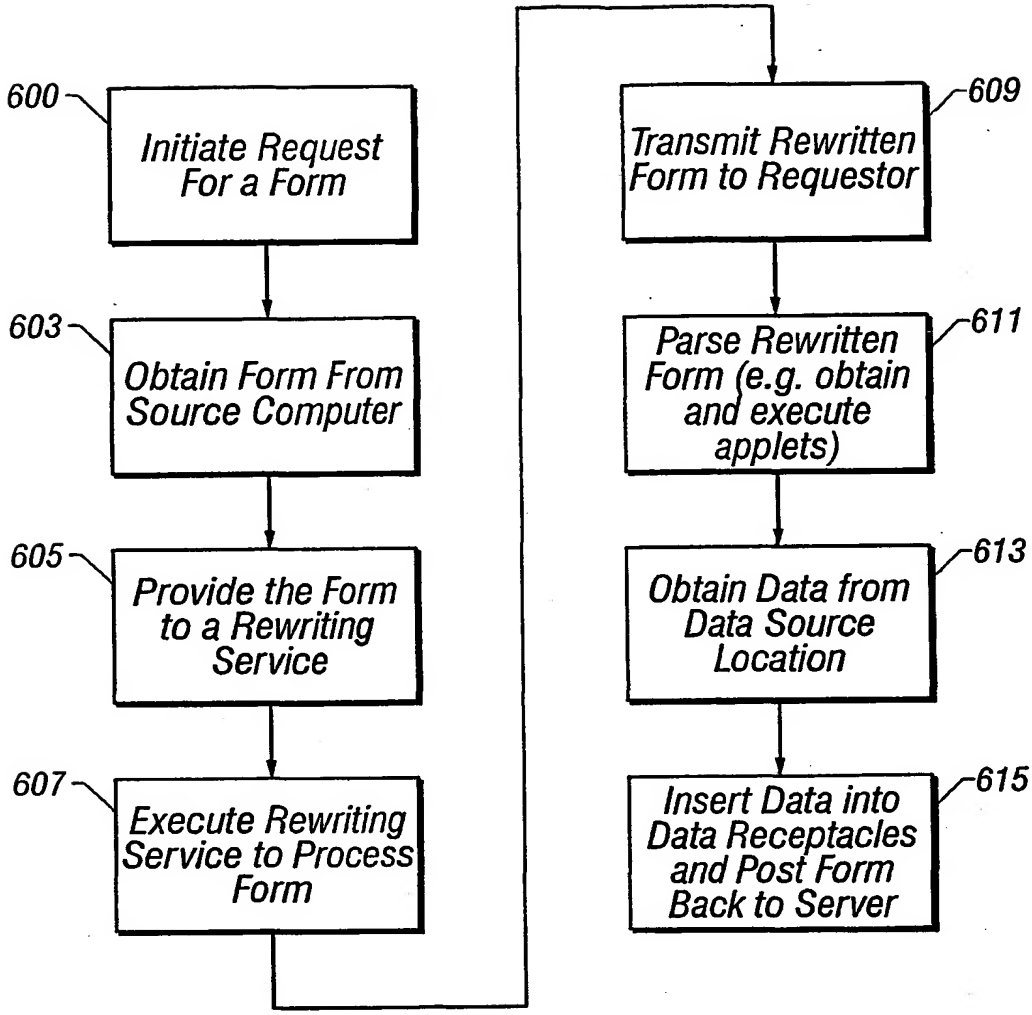


Figure 6

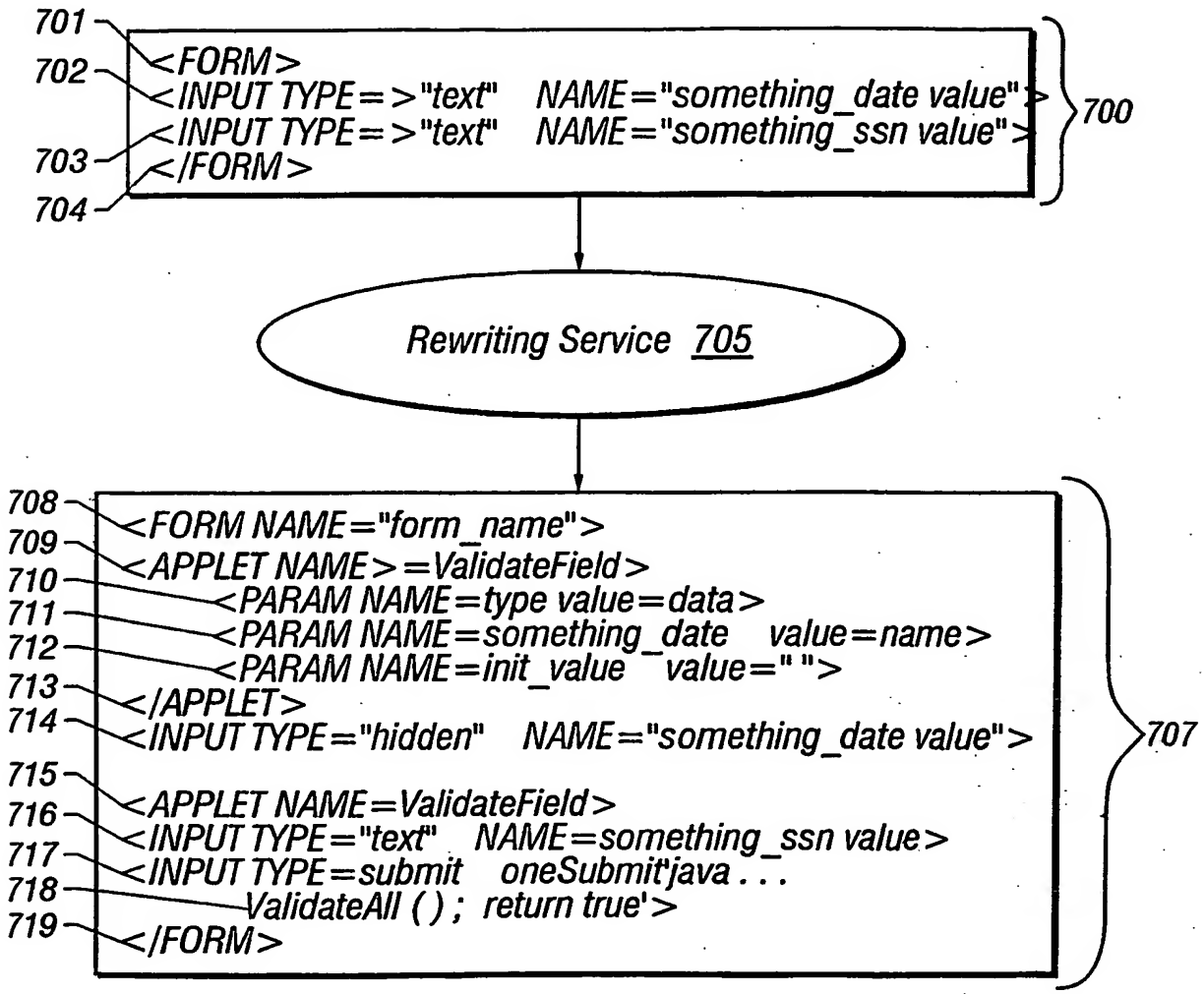


Figure 7

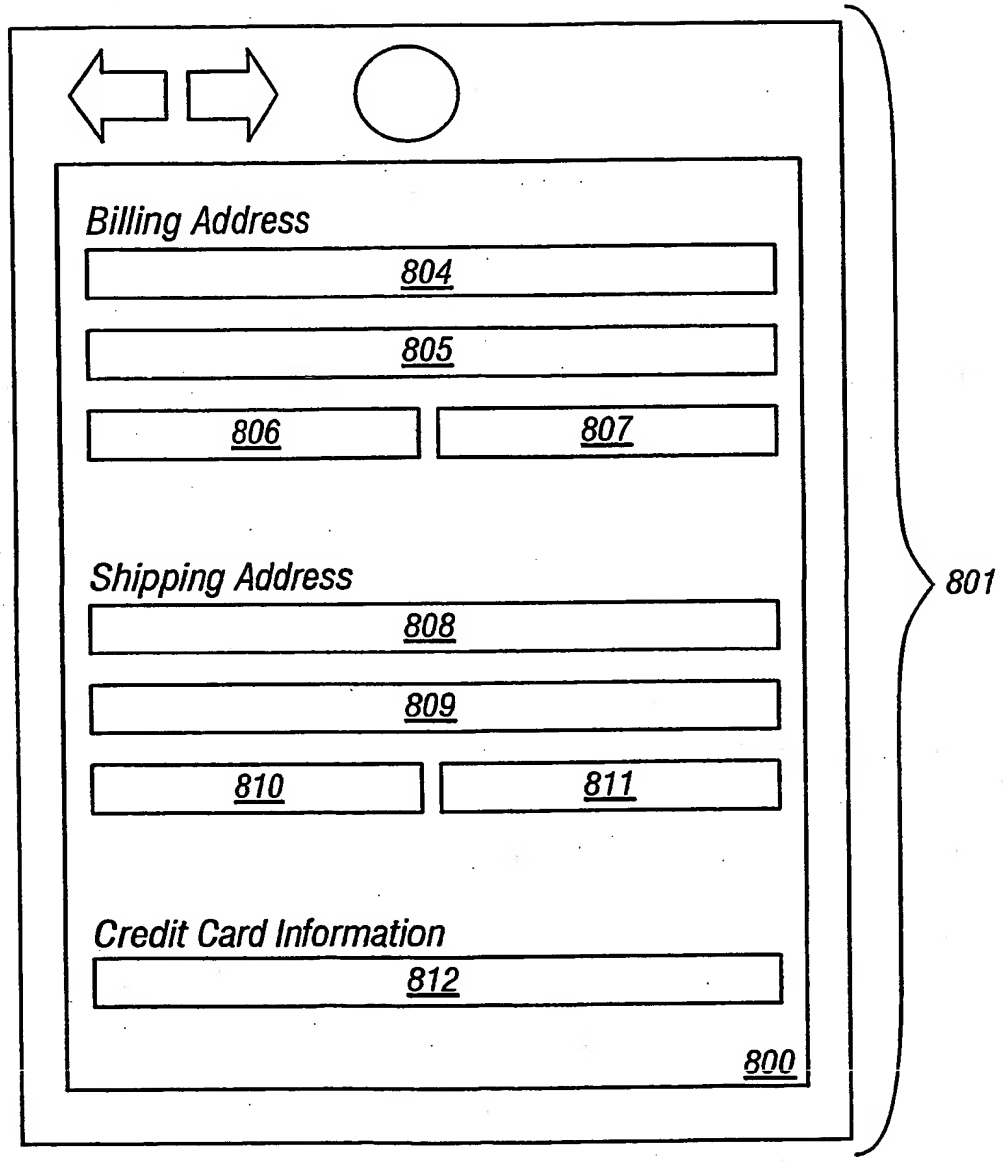


Figure 8